IRI

OASIS + XMOS + Beyond

Software Architecture and Implementation for Digital Reality Worlds, Games, and Other Functions

M. J. Dudziak

version 1.00.00 19.November.2018 (init 18.November.2018)

Table of Contents

Abstract 3
Introduction 3
OASIS Fundamental Architecture 3
OASIS is fundamentally a Geometry 3
Implementing the Geometry and the Entities in Software 4
Inter-Level Mechanics as the way OASIS is built and operated 6
Digital World Objects 6
Reference Material 9
CUBIT logics 9
From the OASIS Architecture workbook (August-October 2018) 15
Main Features, Principal Characteristics 15
OASIS Symbolism, Icons, Representations 17
A General introduction to OASIS 17
OASIS as Environment and Ecosystem 18
Primary Activities in OASIS 18
OASIS World Geometry 22
The geography and topography of the Worlds 22
More on the Primary Activities 24
Implementing the Ten Foundational Principles of OASIS in Software 25
More on OASIS System Framework and Taxonomy 27
§ 8.9.1 Parameter Types 44
§ 8.9.2 Module-level Parameter Representation 44
§ 12.1.1 Mirror Types 49
§ 12.4.1 Example: S-Water 51
§ 12.5.1 EVA Module and Function Naming System 51
§ 12.5.2 EVA Modules 52
§ 12.5.3 EVA Functions 52
Extended NOTES and References 55

Abstract

(later)

Introduction

(later)

OASIS Fundamental Architecture

OASIS is fundamentally a Geometry

The OASIS architecture generates and operates digital worlds that consist of 3D objects in a 4D spacetime universe, any of which can be mobile, linked with different sensory and motor functions, and linked with a variety of other objects as groups or as operational units. These associations and linkages can include a variety of media and sensorimotor functions in both digital world (DW) and physical (real) world (RW) environments. The whole system is designed to accommodate multiple objects, with mobility, in geographic settings, with topographical features for a natural environment and constructive features pertaining to buildings, roads, and other structures. Within these worlds users can perform a variety of actions that are summarized as "COMEET" – Communicate, cOllaborate, Make, Educate, Entertain, Trade.

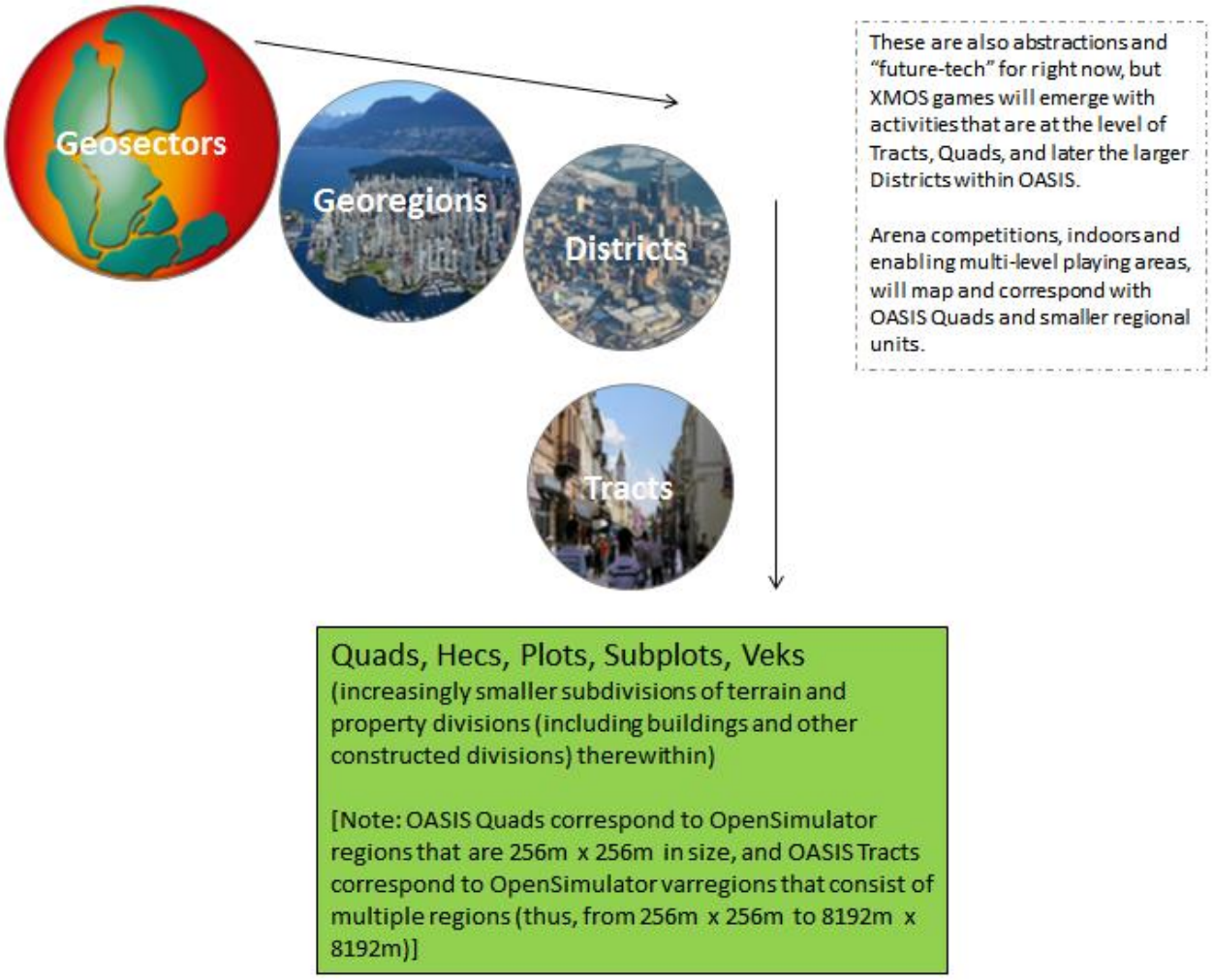
This is essentially all GEOMETRY. The "geography" is ultimately Geometry. This Geometry comprises the hierarchical system (superseding what is in the October OASIS architecture workbook, v12-00-01, from 08.oct.2018) that is presented in iri-oasis-xmos-games-workbook-1_mjd_14nov18.ppt/pdf.

First, for the sake of completeness - on the very large, cosmological, and purely abstract (for now) scale of things, there are these divisions: Sectors → Clusters → Galaxies → Quadrants → Sols.

However, for now and for the most part, everything concerns a more compact scale, that of Worlds. These are where "all the action" takes place in terms of COMEET functions between and among users, players, visitors. Worlds are planet-like, at least abstractly. (In principle, one could have a "flatworld" World that is just a "huge region" and not necessarily a planet-like object).

Within any World we have Geosectors, which are not the same as continents but "more or less". Here is where we will focus now. Within Geosectors are Georegions, which can be like a metropolitan area, a large city or a large rural region. Within Georegions are Districts. In terms of urban areas, for comparison, Districts are like (for example): Lower Manhattan or Queens (NYC), or Soho (London), or central downtown Moscow, or Santa Monica and Venice Beach (LA). Districts are flexible and their boundaries evolve over a period of time based upon user activity and decisions.

But all of OASIS from the standpoint of internal data structures and a database, and the ways that any software will interact at the OASIS system level, is in terms of GEOMETRY and this means coordinate-space (x, y, z) and also time (t).



Implementing the Geometry and the Entities in Software

Everything within an OASIS space (World, District, tract, etc.) is an Entity or ent. This applies to purely geometrical objects, to abstract things like a defined area of real estate or a part of some "natural feature", to buildings, vehicles, avatars, signs, utility lines, trees – anything that can be treated digitally as an object which has some functions pertaining to placement, display, other sensory features, any actuator (motor) features – any behaviors including simply "being there".

Ents are what make up OASIS and its activities. They are implemented in a hierarchical level in terms of the logic, but ultimately, in terms of what happens in a computing device and has results on screens, headsets, audio equipment, and anything else, is happening through a wide and open-ended variety of software from many sources.

OASIS is an open-architecture system that is very platform-independent, multi-application, multi-language. There are three main levels of software:

[M-level] Meta-object, meta-function processes and logics that apply to ents at the conceptual level of ents, and in groups

[O-level] Object and function definition – all the basic characterizations of attributes and behaviors of the ents, in terms of logical objects and their properties

[F-level] Function implementation – the code that performs all the different actions for the objects and their properties

M-Level

The meta-object and meta-function processes and logics are done through software tools particularly designed for Parallel Distributed Processing (PDP), Artificial Intelligence (AI), Network Computing (NC), and Functional Programming (FP). These processes will be developed in the future, since they are not necessary in early stages, and their definitions will depend upon the gradual evolution of OASIS geometries, ents, and the objects and functions that make up those ents. Among the languages-tools that will be used are Occam-pi, different standard and common versions of Lisp (C-Lisp, Clojure) and perhaps other functional languages (e.g., Haskell).

O-Level

Object and function definition is done using the OpenSimulator operating environment, and the script languages that are designed for OpenSimulator – LSL (Linden Simulator Language) and also C# scripts.

All the action in terms of what happens digitally, through some computer software processing, at some OASIS geometrical coordinate(s), is implemented at the First Level within OpenSimulator. This is the high-level and most fundamental software protocol that is used for everything that can be "placed" (grown, built, moved, operated) within any part of OASIS – in terms of DW – Digital World objects and functions.

OpenSimulator makes use of the logic and framework of LSL (Linden Simulation Language) which is also known as the "Second Life" scripting environment.

So here one enters into the level and language of prims (primitives) and everything that can be done by and with objects.

F-Level

Function implementation is done through these scripts and through a large variety of other tools and languages. These include all of the following as available tools that can be used in the development of scenes, animations, and especially applications such as Games and Simulations.

LSL and C#
A-Frame (HTML-type script for Virtual Reality objects and actions)
Unity and unreal engine (major software tools used in gaming and VR)

Blender, Maya, AutoCAD (3D objects, texturing, visualization)
and others, because in the OASIS architecture, there are ways to enter and leave, in the experiential sense and in terms of software APIs, from the basic OpenSimulator implementation of different objects that make up the ents, into applications that may be, for instance, a game that is written in Unity or Unreal Engine.

Inter-Level Mechanics as the way OASIS is built and operated

Thus, we can say that any OASIS location, as a set of coordinates (point, line, polygon, polyhedron) or any semantic representation (street, building, house, lake, river, garden, train, vehicle) is all going to be represented in terms of DW operations by objects that include primitives and the attributes (properties) associated with them, some of which are other objects.

From an OASIS entity there are pointers to LSL primitives and other objects. And we will discuss how we work with those, further below.

The OASIS entity can also have pointers to Real-World (RW) entities. These are also real associations, but obviously different since they are not something "inside the program", digital, represented in software, etc.

The RW objects obviously cannot directly change from anything being done in the DW with software (e.g., OpenSimulator or anything else used or called from such functions). However, there could be situations where the OASIS entity interacts with the RW object(s) in a variety of ways, actively and/or passively. For example, to obtain sensor inputs which could include video or audio. For example, to operate some robot or other servo-actuator device that makes changes in the RW object/environment.

For now, we just keep all this in mind. An OASIS entity ("ent") is located ultimately in one or more "vek" coordinates, and this has some connection at least, minimally, with DW objects (a polyhedron, some combination of polyhedral that can move, etc.), minimally a plain flat surface of finite dimensions, and it may have connection with some RW objects (e.g., a physical house, automobile, train, tree, mountain, waterfall, animal, human).

Digital World Objects

Now to the DW objects.

First of all, there can be multiple DW objects associated with an given OASIS entity, in any finite or complete set of behaviors of that entity (locations, actions). So, multiple DQW objects can be mapped to an individual OASIS entity.

For now, let's stay simple and consider just singular DW objects associated (mapped) with a given OASIS entity.

(And also, the same for RW objects and mapping to OASIS entities.)

What can we do with the DW objects, and how do we operate with them?

This is where it gets very interesting, seemingly complicated, and certainly very powerful.

ent = OASIS Entity

ent.geo = data + code structure for the Geometry

ent.cubit = data + code structure for the Logic Processes that can be done by and in the ent
(CUBIT = CUBIT = Cybernetic Universal Biological Intelligence Technology, using the original semantics (2012+), but also, simultaneously, it incoirporates the 2018 semantics (Constructor for Understanding and Building Intelligent Technology)
This includes everything from the CUBIT taxonomy with cybots and mentats, and other agents and functions, etc.
All of the ent.cubit code and data is in terms of OASIS functions, behaviors, locations, geometries, actions.

ent. dw = everything pertaining to digital objects and actions associated with the ent

ent. rw = everything pertaining to real-world objects and actions associated with the ent

What is in the ent.cubit interfaces with the ent.dw and ent.rw.

Within the ent. dw, we have
ent. dw.os = all the OpenSim (OpenSimulator) representations and code that governs everything that can be at, in and functioning as part of the ent.

All of this interfaces with a very wide variety of code and data which is where we enter into world of

LSL, A-Frame, Unity, UnrealEngine, Maya, Blender, and everything else that can be software for representing, visualizing, animating, and otherwise operating something that comprises the ent with its defined properties and functions as specified in

- ent.geo
- ent.cubit
- ent.dw
  and with regard to interactions that are specified by
- ent.rw

Ent – something that can be something, do something, have appearances, actions, and interactions, within OASIS.

Ents have locations – static or dynamic. Well, “static” is an exception, an extreme. In principle, everything can move and be relocated.

Just as all numbers can be considered to be infinite, to be ir-rationals, but they can be “rounded up or down” to be rationals. In the “exceptional cases” they are so “rational” that they are integers. Something like that.

So something that is really stable, unmoving, permanent, static, is like an integer in a world of non-integer rationals and irrationals…

Ent.geo describes everything about the “where” in terms of OASIS coordinate-space.

Ent.cubit describes everything about the “what” and “how” – everything that can happen, be done, with and by the ent.

Ent.dw decribes everything about the digital world functions and relations, and ent.rw for the real world, the same, the counterpart.

Ent.dw.os contains everything about how the dw functions are done, at the first-level, the high-level, within the language (synax) of OpenSimulator.

But this points to things in LSL scripts and C# scripts. So now we are into actual software code.

But SOME of these ent.dw things , through ent.dw.os interfaces (“API”) will lead to OTHER CODE that is not LSL or C# scripting.

Here we get to A-Frame, and to other code such as Unity, UnrealEngine, and to 3D, CGI, texturing, visualization, animation techniques that have OTHER SOFTWARE besides LSL and C# scripts.

So we are pointing FROM the ent.dw to some software.

And this could be to an entire system, such as a GAME program.

Thus, we can “jump” (and visually, experientially, using some type of graphical “portal” experience), from an ent in OASIS to some OTHER PROGRAM like a Game, which has its own complete “world” and its entrance/exit points.

Then when we leave that OTHER (e.g., the Game), we come back either where we left OASIS or perhaps to some other location in OASIS. There is no reason why we must always return to the same place (or time!).

Reference Material

CUBIT logics

CUBIT Material – for an APPENDIX, but it is here now

CUBIT Taxonomy – a classification system

The historical basis for CUBIT and all the variant classes of cubits, for today, is the original CUBIT architecture designed, developed and applied to different systems during 2008-2015 and in particular the work during 2013-2015.

The following is here as a Reference and what is in the main body of this document takes prece3dence. Do not get confused.

(**CUBIT** /* Physical & computational objects and composites (structures) that act as cooperative and competing agents and as communities of agents, forming collective and dynamic semi-autonomous systems. Such a system, also known as an ASIM (Adaptive Synthetic Intelligence Module (assembly, machine)), employs one or more cubits (instances of CUBIT) to perform a wide range of computational and/or physical tasks. Some cubits serve directed purposes in the "tetrad fields" of "EEHS" (energy, environment, health, security) functional areas, and among such cubits, some then become commercial products).

Formally, CUBIT = Cybernetic Universal Biological Intelligence Technology. The concepts of cube, block, and composition element are important, as are thinking in terms of both silicon-based and carbon-based physical implementations, and logical "molecular modules" in terms of programmatic (algorithm, software) implementations.

An important point is that cubits may be purely logical entities, embodied in logic that is expressed as software codes, or physical entities that are expressed in a variety of micro/nano devices comprising electronic, electromechanical, and molecular (including biomolecular) components. A significant body of cubits are software components and modules that operate with, upon, and within physical-device cubits, but also on diverse other physical platforms (e.g., conventional, classical microprocessors. A body of logical-entity cubits comprises a defined category of software treated as a unitary body, LUX.

In short, some cubits are the logics that control and operate other cubits which can or must be physical devices. */

(**cubit-element** /* **cubit-ε** --- fundamental cubit element */)

(**cubit-structure** /* **cubit-σ** --- fundamental cubit structure, a composite of two or more cubit-ε elements */)

[several orders within the class **cubit-σ** and many families within the order **cubit-λ** *aka* logic-cubit]

(**LUX** /* Computational environment for the operation of cubit processes, consisting of an integrated system of software modules that constitute an "operating system" with a wide and open-ended set of functions for all types of cubits. This "operating system" or "functional processing environment" is designed to be compatible with multiple computing platforms including those that are commonplace today (e.g., Windows, MacOS, Android, iOS, Chrome, Linux, Tizen). The LUX code environment operates as a network of applications that have interface modules for running code on the various hardware platforms within the various conventional operating systems.

Within the CUBIT taxonomy, LUX is a family of cubit within the order **cubit-λ** within the class **cubit-σ**)

LUX provides a semantics-based logic for cubits that inhabit and perform processes in a variety of computational worlds that can be variously described as "real-world" or "virtual". LUX provides the logic by which cubits can perform tasks and interact with one another in LUXworlds and in non-LUX environments. LUX software is used within many types of cubits (both "simplex" and "complex"), including those which are or may be employed within LUXworld-centric experiential environments (VirtuRealities, such as ORBIS) */

(**lux-element** /* **lux-ε** --- a unitary functional module */)

(**lux-structure** /* **lux-σ** --- a composite functional module (application, program) comprised of lux-elements including elements that provide gateways and interfaces to other non-LUX software */)

) /* LUX */

) /* cubit-structure */

) /* CUBIT */

) /* CoAD */

) /* ABORINT */

Note that in previous versions of this logical taxonomy, CUBIT and LUX were parallel phylum-level categories and this has changed significantly. LUX is now defined as a subcategory within CUBIT, specifically a family within an order within the class of cubit-structures. This is significant and hopefully clarifies and simplifies may things.

Now we examine CUBIT in greater depth, for what are (and will be) the classes of cubit-elements and cubit-structures. Many of these lead directly into TetraDyn products, commercialized and commercializable, and for which there is IP that needs awareness and protection under patent law or other measures (e.g., “trade secret” measures).

Then we examine the LUX category (subclass of cubit-structure) in greater depth.

At the next-highest levels we find a close parallelism between the classes and subclasses of CUBIT and the LUX family of logic-cubits which are implemented in software code. Both share qualities of being purely computational or a composite of computational and physical. Both work with each other as logical, complementary, and natural components of systemic applications for highly diversified uses serving both consumers and organizations.

[2] Abstraction Level 2a - CUBIT

Here we examine CUBIT in greater detail.

(**CUBIT** /* physical & computational objects and composites (structures) that act as agents within collective and dynamic systems... (phylum) */

(**cubit-element** /* **cubit-ε** --- fundamental cubit logic element (class) */

(**monad** /* elementary process unit – computational and physical components (order) */)

(**binar** /* elementary agent unit – computational and physical components (Binary Intelligent Neural-like Adaptive Recognizer; Biological-like Intelligent Neural Adaptive Recognizer) (order) */

(**cybot** /* generic, limited-autonomy, limited-modifiability binar – computational element (analogous in some respects to “bots”) (family) */

(**cybernaut** /* a type of cybot dedicated to coordination and control functions within a cubit-structure (genus) */)

(**cybereng** /* cybot dedicated to energy, power, and mechanical functions within a cubit-structure (genus) */)

(**cyberent** /* cybot dedicated to environmental data acquisition and effect functions within a cubit-structure (genus) */)

(**cybermed** /* cybot dedicated to biological and medical-type functions within a cubit-structure (genus) */)

(**cybersec** /* cybot dedicated to security and countermeasure functions within a cubit-structure (genus) */)

) /* cybot */

(**mentat** /* self-organizing and semi-autonomous binar – computational element with greater complexity than a regular cybot (analogous in some respects to “bot masters” - subtypes include same classification order as cybots (family) */

(**m-cybernaut** /* a type of cybot dedicated to coordination and control functions within a cubit-structure (genus) */)

(**m-cybereng** /* cybot dedicated to energy, power, and mechanical functions within a cubit-structure (genus) */)

(**m-cyberent** /* cybot dedicated to environmental data acquisition & effect functions within a cubit-structure (genus) */)

(**m-cybermed** /* cybot dedicated to biological and medical-type functions within a cubit-structure (genus) */)

(**m-cybersec** /* cybot dedicated to security and countermeasure functions within a cubit-structure (genus) */)

) /* mentat */

(**kyber** /* type of binar specifically dedicated to KYBEROS hyper-encryption and hyper-security tasks (family) */)

(**soph** /* higher-complexity, higher-autonomy binar – essentially an assessor, evaluator, and coordinator of mentats and cybots and elemental binars – subtypes include the same classification order as with cybots and mentats (e.g., cybernaut... cybersec) (family) */)

) /* binar */

) /* cubit-ε *aka* cubit-element */

(**cubit-structure** /* **cubit-σ** --- fundamental cubit machine structure – note well that “machine” may be purely logical and non-physical (e.g., program code) (class) */

(**cubit-λ** *aka* **logic-cubit** /* algorithmic, logical, computational structure – manifest in software, not in physical components, although the logic may be implemented in firmware such as a micro/nano-processing object (order) */

(**binet** /* configuration of logical, computational operations and processes constituted and conducted by binars; a binar community, colony, organization; this may be distributed over a network of devices including servers, standard computers, mobile devices such as phones, tablets, koins (qoins), etc. (family) */)

(**cyborg** /* organism/community-configuration of binets – example: as implem ented in a magicCircle; same remarks as above regarding physical distribution (family) */)

(**cyberecos** /* ecosystem-configuration of cyborgs – example: as may be implemented in a LUXworld ; same remarks as above regarding physical distribution (family) */)

(**LUX** /* computational configuration (environment) for the control and operation of cubits in applications involving interactions and interfaces with conventional computing devices and humans; the configuration may be widely distributed as above, such as in a multiplicity of devices within several distinct networks as defined by those devices and their conventional operating systems (family) */)

) /* cubit-λ *aka* logic-cubit */

(**cubit-μ** *aka* **cubit-Mod** /* physical, mechanical, modular device – e.g., micro/nano-processing object, MEMS device, electromechanical machine, module composed of several integrated circuit components, simplex or complex cubit physical device, macroscopic machine, etc. (order) */

(**cubit-δ** *aka* **simplex-cubit** /* simpler, unitary, modular “building-block” type of physical cubit that can be combined physically and functionally with other simplex and complex cubits (family) */

(**cube** /* physical cube-geometry cubit (genus) */

(**C-cube** /* computation cube (subgenus) */)

(**D-cube** /* data-acquisition cube (subgenus) */)

(**F-cube** /* function cube (subgenus) */)

(**I-cube** /* interface cube (subgenus) */)

(**M-cube** /* memory cube (subgenus) */)

(**P-cube** /* power cube (subgenus) */)

(**R-cube** /* router cube (subgenus) */)

(**T-cube** /* transform cube (subgenus) */)

) /* cube */

) /* cubit-δ *aka* simplex-cubit */

(**cubit-κ** *aka* **complex-cubit** /* non-simplex, non-cube, variational design and function, compound-element cubit – a cubit with multiple components, some of which may be other cubits (family) */

(**cebit** /* **c**hemical, **e**nvironmental, **b**iological **i**nteraction **t**echnology unit – a specialized F-cube or multi-cube device that serves as a sensor or actuator designed to react with one or more specific stimuli of inorganic or organic chemical compounds including biologically-active and radioactive substances (formerly treated as two categories: CEBIT = **C**hemical, **E**nvironmental, **B**iological **I**dentification and **T**racking, and CEBRA = **C**hemical, **E**nvironmental, **B**iological **R**esponse and **A**ctivation) (genus) */)

(**qoin** /* a physical cubit, in different shapes and sizes – a device that is coin-like in function and potentially in appearance, serving as a repository of data and code for enabling, protecting and processing that data, with a number of wireless communication and power functions. Alternative and prior-dominant name: koin.

Also known as a knowledge object intelligence node – a device that is coin-like in function and potentially in appearance, serving as a repository of data and code for enabling, protecting and processing that data, with a number of wireless communication and power functions, with implicit economic value in both “real” and “virtual” terms, and typically carried, shared, and traded by persons for other koins.

Each qoin (koin), regardless of physical geometry or composition of the primary materials, contains minimally: non-volatile and volatile memory, passive RFID, wireless communication (e.g., bluetooth), microprocessor, and a physically unique identification ('fingerprint”); qoins may be assembled into cooperating processor arrays by one individual or by a group of individuals as long as the qoins are in sufficient proximity to one or more devices that can serve as the main processing and intercommunication device (e.g., desktop, laptop, tablet, or smartphone computing machine).

Note that there are four basic types (species-level) of qoin – iqoin (ikoin), pqoin (pkoin), mqoin (mkoin) and xqoin (xkoin). In the case of pqoin devices, there may be a combination of a cebit device as a module within the pqoin, mainly where the sensor or actuator functions demand a more complex and modular architecture. (genus) */

(**iqoin** /* interactive/control qoin – multi-protocol communications, memory, decorative-skin-capable (species) */)

(**mqoin** /* memory-intensive qoin – memory & power enhancements for use with other devices (species) */)

(**pqoin** /* process/function qoin – sensors, actuators, and other specialized processing (species) */

(**pqoin-α** /* pqoin that is a singular-construction non-modular device, all electronics embedded at time of manufacture within the core shell of the device (subspecies) */

) /* pqoin-α */

(**pqoin-β** /* pqoin that is a modular-construction device, containing electronics embedded at time of manufacture within the core shell of the device, as well as a removable or changeable module that is inserted or attached to the device. (subspecies) (variety) */

) /* pqoin-β */

) /* pqoin */

(**xqoin** /* generic/basic qoin – no pre-built electronics but capable of add-ons and inserts (species) */)

) /* qoin */

) /* cubit-κ aka complex-cubit */

(**cubit-θ** *aka* **cyberMod** /* composite structure of one or more cebits and cubes for specific ranges of functions, such as medical diagnostics and therapeutics; energy monitoring, production, conversion and storage; environmental analysis and forecast; physical and cybersecurity measures and countermeasures; leisure and entertainment; systems control and cybernetics (family) */

) /* cubit-θ *aka* cyberMod */

) /* cubit-μ *aka* cubit-Mod */

(**cubit-π** *aka* **cyberPod** or **cubitPod** /* physical, mechanical Pod-type device – e.g., human-sized, walk-in, portable room-structure, in which a variety of subsystems are installed, some of which may be cubit-instances, but most will be non-cubits (family) */

(**nPod** /* physical macro-structure Pod (Purposive Operational Design), composed of Pod-specific components including structural frame-elements; structural frame-elements (SFE) are composed of combinations built from what are called Base-Elements (“BE”), such as rods, tubes, and cables, with Joiner-Elements (“JE”), devices that serve to link two or more BE into a structural element (SFE); other nPod structural elements vary according to the type of nPod (genus) */

(**nGon** /* basic modular nPod built from a configuration of SFE using a standardized template for faces, walls, floors, ceilings (subgenus) */

(**nRec** /* nGon assembled as a rectangular prism from basic SFE – the most common type of nGon and the primary building block for all PodAtriums and most non-Space (non-ETA (ANKI)) cyberPods within CUBIT-Pod (species) */)

) /* nGon */

) /* nPod */

(**PodAtrium** /* (“multiPod”) composite structure of one or more nPods for specific ranges of functions, such as a medical mini-clinic, environmental field testing lab, emergency response facility, and other applications (genus) */)

) /* cubit-π *aka* cyberPod or cubitPod */

) /* cubit-σ *aka* cubit-structure */

) /* CUBIT */

[2] Abstraction Level 2b - LUX

Here we examine LUX in greater detail.

(**LUX** /* family of software architecture and modules comprising an operating environment and programming language for Adaptive (Communicating) Dynamic Processes – processes that may operate in isolation, in MIMD parallelism, in community-like structures and hierarchies of operation, competitively, concurrently, and which have the capability of self-modification of code, through the logic of binars (family) */

(**ESX** /* software modules comprising a toolkit and APIs for design and construction of cubits – both cubit-ε and cubit-σ types (genus) */

(**APIS** /* Anomaly and Pattern Interface Schema – subset of ESX - software modules designed to provide functions that can be employed by binars in a variety of pattern detection, recognition and learning applications (subgenus) */

(**hive** /* software construct/assembly using APIS modules and dedicated to a specific set of cognitive tasks (species) */)

) /* APIS */

) /* ESX */

(**iQs** /* aka **QOIN -** Knowledge Object Intelligence Network – software modules for managing QOIN-integrated data objects and transactions among network members/users and including but not limited to functions embedded in physical koins (qoins). Also known as KO'IN, KO-IN, (formerly KOIN), IntelSphere, iQsphere (intelligent QOIN sphere/space), iQs (genus) */

) /* iQs, QOIN */

(**LUXspace** /* abstract general form of a LUXworld; also referring to the collection of all LUXworlds (genus) */

(**LUXworld** /* software modules constituting navigable, world-like map-enabled environments that interact with and serve as bridges between “real world” and “virtual/game world” systems; a “VirtuReality” world (subgenus)*/)

(**OpenNet** /* instance of a LUXworld that is a stable bridge-platform for interfacing multiple, parallel LUXworlds – comparable to a rail station that serves multiple lines, each of which operates multiple trains or different types and schedules (variety, instance) */)

(**T'rain** /* (“Terrain”) instance of a LUXworld that is a stable environment for interfacing multiple geographies and topographies among “real world” and “virtual/game world” systems (variety, instance) */)

(**KOINet** /* unique one-of-kind instance of a LUXworld comprising all data and metadata communications involving qoins (koins) (variety, instance) */)

(**KyberHaven** /* (“Hades”) unique one-of-kind instance of a LUXworld that enables the implementation and management of the Kyberos encryption network-cloud and the operation of the iBank business through this exclusive structure (variety, instance) */)

(**ORBIS** /* Singular public-access (multiple subscription levels – free and paid) VirtuReality world that is an environment integrating artificial/synthetic features into the natural/contemporary world, for gaming, trading, and entertainment, incorporating many of the features originally featured in fictional worlds such as Terrain (T'Rain) in Stephenson's “REAMDE” (variety, instance) */)

) /* LUXworld */

) /* LUXspace */

) /* LUX */

From the OASIS Architecture workbook (August-October 2018)

Main Features, Principal Characteristics

These are the main features, rationales, and other principle points about OASIS.

OASIS is something very new and different from anything previously in the main disciplines and technology areas that are involved, including all of these: computer science, cognitive science, machine learning, database and data mining systems, internet applications, social networks, virtual reality, artificial worlds, simulations, games, visualization, user interfaces.

All of the aforementioned branches of STEM are in some manner components and ingredients within OASIS, but the integration and synthesis includes different motivations, different orientations, different foundational thinking and engineering.

Foremost among OASIS motivations and goals is that is it an environment for people to do constructive, interesting, useful, productive, pleasurable and profitable activities that, in the process of everything being undertaken, will strengthen many important attributes and qualities for people as individuals, groups, communities and as a whole society.

These paramount and essential objectives include:

- personalization and direct interpersonal engagement among people, in contrast to depersonalization and mechanization,

- critical analytical thinking and cultivation of rational, scientific methods,
- discernment between reasonable accuracy, speculative possibility and deliberate misinformation,
- imaginative and creative thinking, designing, and implementation in different artistic, scientific and technical areas,
- overcoming of cultural and meme-based barriers and constraints,
- freedom from harassment and intrusion through overt spam, advertisements and other unwanted and unsolicited media
- provisions for privacy and freedom from external third-party capture or control of personal data, extending to anonymity of identity and content

The OASIS environment is not only something employing computers, internet, and information technologies. OASIS is both informational and physical, virtual and real, online and offline. This is a very important, fundamental, powerful and valuable difference from everything else that has been invented, engineered and offered – both commercially and non-commercially – using the internet and computers, since the very beginning of the "information/computer age."

To accomplish these objectives and to implement these functions, OASIS makes extensive use of games, contests, competitions, prizes, surprises, and many elements of attraction and sustained activity that has elements of leisure, surprise, chance, and skill. It is not purely an MMORPG environment, but it uses all the features found in games, both 1:1 and Many:Many. It is not purely a "virtual reality" world available only on the internet, but it uses such technologies and offerings.

OASIS functions use but do not depend solely or focus exclusively upon internet, virtual reality, artificial intelligence and other technologies, especially those emphasizing data and its transformation into information. The worlds within the OASIS universe all function through the internet and many computational, electronic (digital and analog) technologies, but there are also features, and capabilities for development and extension by users in the OASIS communities, for very physical, on-the-ground, nuts-and-bolts components. All of this is very much in the hands of the members, the users, the communities, enabled and assisted by a large and open-ended, ever-growing, always-innovating set of tools, resources, and people.

OASIS is an environment for both personal and group activities, non-commercial and commercial applications, and it serves all aspects of the following six basic Core Function categories:

Communicate – between individuals and among diverse groups from small to very large and open-ended communities. The focus is upon making point-to-point communications easier, more secure, more comfortable, and more personable.

Collaborate – also between individuals and groups, for diverse activities ranging from leisure and entertainment to education to research to all aspects of technical and non-technical adventure, study, and commerce. The focus is upon making activities more conducive to sharing ideas, workloads, responsibilities, and making it easier to do what really matters within the group activity, with less overhead of "project management" and "technical dependencies."

Educate – exploring, learning, studying, training, as individuals and groups, in a variety of subjects, not only but especially those classed as "S.T.E.A.M." - science, technology, engineering, the arts, and mathematics. The focus is upon bringing into the learning experience the genuine feeling and practical value of being actively connected

and involved with some project (a research group at a university or company, for instance, or a production facility) that uses the knowledge being acquired, to do something practical and tangible.

Make – inventing, prototyping, producing devices or structures, singular or in mass production, not only geometrical or mechanical, but virtually anything, and not only using certain technologies such as "3D printing." Things may start out "virtual" but the intent of the "Make-ing" here is to produce something that can be used by someone.

Trade – all forms of individual and institutional banking, bartering, buying and selling, and trading of both specialty objects and products, commodities and futures, stocks and other securities. The focus is upon people making profits, and this includes tie-ins with established banking and trading houses. However, there is also something special for within OASIS, a type of crypto-security, and also there is the private information banking and trust services (I-Bank).

Play – all forms of individual and multi-player gaming and some things that are unique to the OASIS environment. The focus is upon making games and leisure be compatible and supportive of profit-oriented trading, making, and collaborating.

"CCEMTP" - it does not make an easy acronym, but these are the six major activities in human social life and in OASIS all of these can be accommodated.

OASIS Symbolism, Icons, Representations

OASIS is closely linked with the CUBIT architecture of knowledge representation and constructor-assembly components. Naturally the cube enters into the picture in terms of symbolism. A cube is one basis symbol used in OASIS, and its six faces pertain to the six fundamental functions as introduced above. Likewise for the hexagon with its six edges.

Polygons and polyhedral are important structures and symbols within OASIS. However, there is no limitation to how users, in creating their new worlds and features therein, can make their structures and the symbols for representing them.

The concept and mechanism of KOIN (from TetraDyn and Tetrad Group, developed first @ 2012-2015) is something to be considered and used in OASIS. See http://koin.tdyn.org. (Koins can also be considered in relation to Cues and Cubits in the Science Communications ("SC") developments between Mirnova and Candeed Cue.)

The OASIS Universe (described further below) is symbolized often by a torus, and the different Worlds by spheres, but also Worlds can have their own distinctive, unique icons, and they are generally geometrical and based upon sphere-concepts, but not necessarily limited to that geometry base.

A General introduction to OASIS

All of these activities take place within the OASIS worlds. These begin and remain and grow as digital worlds,

with a variety of standardized and modifiable geometries, physics, ecosystems, landscapes, topographies, and constructions that vary according to how the builders, owners, visitors, and others interacting with these worlds choose – according to basic logics of engagement and control – with these worlds and with each others as co-creators, owners, inhabitants, and otherwise as users.

Within each OASIS world there are the capabilities for many functions, many constructions, and all many be summarized broadly as Communicate, Collaborate, Educate, Make, Trade, Play.

What do we do as humans, as a society, from time immemorial? We communicate with each other. We collaborate in different ways. We learn, train, and educate. We create, make, construct, and build. We trade through barter, exchange, buying and selling, and investing. We play, entertain, recreate. Naturally, everyone has different meanings, different associations and uses, for all of these activities.

OASIS as Environment and Ecosystem

OASIS is an environment for enhancing personal and social interactions on many levels, using the functions provided through computers and other devices with the internet and other means of electronic communication. Its functions may at first appear to be similar to many things available through other products and services. However it differs very strongly and significantly from much of what has surfaced and been experienced, to date, in the manner of social networks, content management systems, virtual reality, online gaming, online trading, and other offerings.

OASIS is intended and designed to cultivate, to promote, to deepen the human experience of communication and group activities. It is an environment for increasing discovery, learning, knowledge, creativity, invention, innovation. It is an environment for deepening and enriching human interactions with other people in very direct ways including physical, face-to-face interactions. And in all of OASIS activity there is the capacity and resources for privacy, anonymity, independence, and freedom from intrusion such as we see today from many social networks and other internet providers.

OASIS is not simply a virtual reality (VR) world. It includes functions for the use of VR as well as AR and MR media, and there are “worlds” which are central features within all OASIS functionality, but it is very much integrated with the “Real World” of people, places, events, finances, business, sports, leisure, and basic living.

Primary Activities in OASIS

(These will have additional points here, to be collected and edited from paper notes and other sources.)

The Primary Functions are six (6) in number, as introduced earlier above. There are some core “concentrator functions” within the Primary Functions. They are not to be thought of as “secondary” but more as “specialized” and “concentrative.”

All of these functions are implemented through and make use of the functions and features of the different OASIS Worlds in which these Primary Function activities take place or, if not so directly, then for which there are definite links and pathways connecting World actvity with what people and groups do among each other through these basic functions.

Communicate

Individuals and groups, in the manner of common communications – voice, video, media exchange, and using the protocols and services of common-use, standardized applications (WhatsApp, Skype, Viber, KakaoTalk, Telegram, etc.)

Concentrators:

Meet – built-in tools and aids (services, apps) for Meeting new people, and this is enabled and performed in three basic ways: Active-Initiative, Active-Assistive, and Passive-Assistive.

> Active-Initiative – not so different from how things work in the internet since the "Beginning of Time" and pre-Web, this is where people can actively, directly, on their own initiative, seek out to meet others with whom there may evolve communications and relationships of different types – personal, professional, and "mixed" (both).
>
> Active-Assistive – things are assisted by SI, through cybots (agents) that use SI logics to suggest persons for a "good match."
>
> Passive-Assistive – more is left in the hands of the SI cybot agents and persons go along with their suggestions and then see how things evolve.

Date – this is "Meet" focused strongly on personal, romantic, and implicitly/presumably, sexual-relationship interests.

Aggregate ("Coadunatio") – this is focused upon gathering people together for some purpose which may be to form a functional group, to explicitly Do Something Together in the context of OASIS, or to raise attention and awareness, or attempt to raise funds and other forms of support, etc.

(other, future) - tbd

Collaborate

Individuals and groups, also using common-use, standardized applications.

Concentrators:

These are much the same as for Communication, but with some variances and additions.

Meet – built-in tools and aids (services, apps) for groups to meet new people who are individuals or other groups, for the express purpose of some type of new collaborative activity, and this is enabled and performed in three basic ways: Active-Initiative, Active-Assistive, and Passive-Assistive.

> Active-Initiative – similar to how it is for individuals.
>
> Active-Assistive – things are assisted by SI, through cybots (agents) that use SI logics to suggest groups and persons for a "good match."

Passive-Assistive – more is left in the hands of the SI cybot agents and groups go along with their suggestions and then see how things evolve.

Aggregate ("Coadunatio") – this is focused upon gathering people together for some purpose which may be to form a functional group, to explicitly Do Something Together in the context of OASIS, or to raise attention and awareness, or attempt to raise funds and other forms of support, etc.

Project – this becomes something like a formal, well-organized, and structured Project, where standards of Project Management enter into the process. Here is where online tools such as Slack can be utilized, with appropriate modifications for "seamless integration and utility" including transfer of data into other parts of OASIS that are used by the project members (e.g., things that enter into a world and into specific properties, buildings, objects in that world, constructed and placed by the users).

Trade

- Bartering
- Buying and selling
- Standard currencies and cryptocurrencies
- and
- Information as commodities and securities: I-Bank – Private Information Bank and Trust

There is access, through different places in Worlds, to established, formal, commercial banks, investment brokerages, and other securities trading companies. But there are also the following, all implemented through OASIS Worlds but using open-source content management systems (CMS) packages, which "alpha" user-owner-players (UOP) set up, all following the basis "first come first dominant" natural selection process for such functions within the Worlds.

- Yamarka – a market that has an open-ended but finite number of kiosks (determined by the property size and other constraints) which are assignable by ownership or usually by lease to different sellers. They do their business as they wish, and there are only such kiosks, but they may be large or small. Kiosk operators have their various freedoms about how they do their trading and whether for money or on some barter basis.
- Casbah – a market that is like a Yamarka in many ways but more "fluid" and with many individual traders who do not necessarily have a permanent kiosk operation but may be simply, "in and out" and on an occasional basis, fitting with what they have to trade. They may buy and barter as well as sell for money.
- Auction – a regular auction format, operated by someone (individual, group or company) that has the particular auction license within that World or Worlds.

It is very important to recognize that "real-world" companies will be present in OASIS worlds with their banks, securities brokerages, and stores. There is a place for Amazon, Alibaba and the rest. There is no need for people to build their own market entities, although they are free to do so. Moreover, OASIS will provide the backbone of anonymity and privacy, using private smart contract mechanisms such as Enigma.

One format relationship that is a goal and an example: we want to be the online presence for the Dorotheum of Vienna.

However, there are some unique market types with special-access, restrictions, and benefits. These do interface with at least the OASIS-internal-virtual markets, and "tbd" with established mainstream banks and brokerages. These markets include:

The Mines
(One of which, for reasons connected with a brand of hard cider favored by certain miners, is named the Strongbow Mines)

The Docks
(just like the old dock areas in port cities, like London, Marseilles, Baltimore, Osaka – and yes, seedy and dangerous!)

The Pits
("open-pit" mines and the connotations thereof)

These different markets will be designed by different user-owner-operators. In some cases it will seem like a maximally-decentralized "crowd" and in other cases like a very organized and highly controlled "gang." It all will evolve according to "natural selection principles."

Educate

Courses and seminars and workshops, both formal and informal, but separate and distinct from other functions and also some that are tightly coupled and embedded within games, for instance.

We can make use of extensive tools already existing and easily incorporated into OASIS, such as Moodle, and other webinar mechanisms. With great care we may want to use YouTube and similar video provider tools.

[see Mirnova Academy descriptions on website and in documents]

Make

Inventing, prototyping, producing of goods which may be physical or informational in content.

This is where OASIS as a VR environment links in directly with some fab-labs and maker-spaces. People can create some object and then produce it using, for instance, 3D printing. There are many variants that can be explored and implemented. The design and construction is not limited to physical, mechanical components but can include processes that may be chemical, electronic, or also informational (e.g., software).
This is also an avenue for prototype introduction and marketing. Things produced by whatever means can be visible and available within the OASIS worlds. For instance, a new style of artistic ceramic dishware, or furniture, or lighting, can be distributed, "virtually," among friends, neighbors and merchants in a city within a world, and seen by potentially millions.

Play

Gaming, using the Worlds, and again, this takes us to the "Terrain" or "T'Rain" model. This is a very important and rich area for development. The games can be literally of any types, from tabletop 1:1 games similar to chess or Go, or they may be classic MMORPG games. They may also be such that include both the online and onsite aspects (e.g., proposed "R3G" games).

It does not matter whose game it is, the connectivity is available for linking it in through the experience of an OASIS world. The important point here is that the game may be deeply integrated within the world and various functions (e.g., trading, learning), or it may be something where players "break out" from the OASIS environment and participate in some "third-party" game, but the results, including points won and lost, from playing that "outsider" game can be linked into OASIS for providing values, including points and accrued earnings, for use in other OASIS functions (generally of a commercial value to the player).

All of these functions are performed in a variety of user-decision ways, and the decisions are made by 1, 2, or however many persons or group-entities are in some interaction.

The activities may be done in the simplest of ways both online and direct, and neither VR nor AR nor MR is required – but often it will be desired and preferred, at least for some of the interactions.

This brings us to a second-stage introduction to the OASIS WORLDS.

OASIS World Geometry

Worlds are VR-capable, AR-capable, MR-capable environments designed by members of the community.

Think of them as planets, but there is no pre-defined physical relationship between them.

A World may be based strongly upon an existing place on Earth or any other known object in the physical universe, or it may be completely a fantasy-creation. There may even be a world that is like (even named) *Magrathea* (referring to Douglas Adams' classic novel and series, "Hitchhiker's Guide to the Galaxy").

The geography and topography of the Worlds

The OASIS Universe is divided into Sectors, then Clusters, then Worlds, then Regions, then further. A paramount rule governing all world-type implementation (not only but certainly always involving software and the online digital experience) is that however different components are designed, constructed, and implemented, always it is possible, from the topological and ontological perspectives, to navigate between different places. There may be rules imposed by the owner-operators, but those are figuratively like (or literally, also, like) fences, walls, moats, etc. But in terms of the basic geography, you can move from place A to place B, wherever that is.

The Universe divisions are as follows:

Sector – a cubical unit of 1 s.u. x 1 s.u x 1 s.u where “s.u” is a sector unit-length – this translates to “x” A.U. or light-years (tbd). Sectors are uniform in their geometry as cubes in 3-space of the same parameters.

Cluster – a group of Worlds in one Sector. Clusters do not follow any set rules of geometry. The worlds constituting a Cluster are “close enough together” but there are no specific limits. A cluster will likely occupy a small region of a Sector’s volume, and arbitrarily we may set the maximum average distance or approximate “diameter” of a cluster as being 0.1 sector unit-length.

World – this is equivalent to a “planet” but it is not necessarily following the typical astrophysical description of a planet. It could be an odd-shaped body, and potentially it can be of synthetic origin, constructed by beings such as humans or robots.

Initially, there is one World, and that is known as Terrain, aka T’Rain, aka Terra.

Region – a flexibly defined and delimited part of a World with mainly contiguous boundaries (for instance, something “like” a continent-type region, with nearby islands allowed, etc. It is not a political or socioeconomic entity. Depending upon the World, the number, shape and size of Regions will vary considerably.

District – A reasonably contiguous subsection of a Region, somewhat akin to a “state”-like region where there are some basic features in the world geography that set the district apart, or where the division is by property lines, something that depends upon how the Region is being divided up among owners, leasors and managers.

Tract – an area of property that is larger than a Quad but smaller than a District.

Quad – 200m by 200m, a square of 4 hectares (1 ha = 100m x 100m or 10^4 m2)

Hec – one hectare (called a “hec” simply to have a 3-letter name, more easily distinguishable)

Plot – tbd, but a division of a hec.

Subplot – tbd, but a division of a plot.

Ultimately there is an Elementary Space or “vek” (volumetric elementary containment). This is not just “voxels.” A vek is something that is a basic and not-further-divided (at the moment!) space. It could be equivalent, for a time, to a subplot, plot, hec, or larger division. For the most part it is a smaller component of a subplot, and examples could be:

- garden
- house
- room

But a vek could obviously be divided further, in some arbitrary future time – this must be kept in mind. This is up to the persons owning, renting, otherwise using the vek and that which contains it.

Thus, a vek can contain a number of veks and so on down to some limit-point, but this limit is not so much geometrical as it is pragmatic – how the spaces are being utilized.

These divisions all pertain to the 2D and 3D mapping of worlds.

There are divisions according to whether one is on a World surface or having territory that is below or above that surface.

Surfs – surfaces, regardless of topography – just the surface

Subs - subsurfaces, volumes that extend below a Surf

Supras – suprasurfaces, volumes that extend above a Surf

More on the Primary Activities

Communication

Cultivating personal, direct, real-world communications, using the internet to support, sustain, and augment the "real person-to-person, group-to-group" engagements. Quite the opposite of many internet offerings like Facebook and LinkedIn, from a fundamental philosophical perspective.

Encompassing STEAM and SciComms – art and science engagement, interaction, communications

Also all the earlier KOIN design, product, logics.

Integrating the best tools from the internet:
WhatsApp, Viber, Slack, maybe Skype. Avoiding Google products.

Everything that is "out there" in the "regular world" can be inside OASIS in some way, also. This goes for stores, businesses, all sorts of places. These are all understood as Functions, and these functions may be completely within OASIS or they may be OASIS-presences of existing institutions, companies, services, etc.

Collaboration

There is a goal-directed role to Collaborations – getting people to team up, organize, socialize, and work together. It is the *Coadunatio Factor and Effect*.

People, and their teams and groups, get points, bonuses, other benefits, by working together, by not being "solo" in OASIS, and by practicing "coopertition" – collaboration in the context of competition – with others. There are various rules set up for different programs, for offerings, defining what x can get by being cooperative and collaborative.

Make
Basically, how we are making things in a fab-lab and maker-space way.

Education
Basically, how we are doing and planning things re: Mirnova and especially Cues and Cubits.

There are galleries, museums, theaters, exhibitions, libraries. These may be created and set up by anybody, and this range includes existing, "RW" institutions.

Trade
See everything on I-Bank and Kyberos systems
There are Markets.
There are presences in OASIS of established "real-world" banks, investment brokerages and exchanges, even insurance companies, and certainly of ICOs, both for cryptocurrencies as well as other forms of "private smart contracts."

We want to cultivate trading in some special areas:

- Private Information
- Intellectual Property, both theoretic and pragmatic (including RW inventions).
- Art – paintings, all sorts of works of art.

Play (Entertainment, Leisure)
Focus on "Terrain" ("T'Rain") concepts. The gaming is running throughout the whole of OASIS. Just read about the other functions, and see the ways in which this can be part of a game, and in which a game can serve the objectives for those functions.

Implementing the Ten Foundational Principles of OASIS in Software

These are the ten principles, and following \each is a block of comments about how we do this with the software.

1. Virtual custom personal worlds, much like in the VR/Cyber literature

   xxxxxxxx

2. Cultivating personal direct real-world communications and using internet and computer technology to enhance that. Quite the opposite of the early 21st century social networks and gaming which tends toward reinforced isolation, separatism, and "anti-social, autistic-like behavior." In both implicit and explicit ways, OASIS counters FB, Twitter, etc.

   xxxxxxxx

3. Total Anonymity – private smart contracts. Absolutely the oppoisite of Google, LinkedIn, FB, etc.

   xxxxxxxx

4. Emphasis on education, sharing, coopertition, learing, novelty, structure, concentration, discipline, memory, countering ADHD and "memory-concentration-drift." Cues and Cubits ; "SC-plus."

   xxxxxxxx

5. Life-enhancing, people-helping, socially-constructive and enabling through projects, games, trading, works of all sorts. (Fits with Christianity, Buddhism, Dharma, L.I.F.E., others.)

   xxxxxxxx

6. Barter-sharing personal-family business models. (e.g., like families and neighbors do in different communities – Midwest USA, Scotland, Tenerife, etc., and like the "AIOUE" model of 2012+)

   xxxxxxxx

7. Enhancing and training and teaching and guiding people to be More personal, countering depersonalization, and emphatically connecting people with ways to help each other.

   xxxxxxxx

8. I-Bank and Private Information Bank and Trust services, and an alternate "crypto-plus"economics and finance that transcends "currency", and certainly different from conventional up-to-now cryptocurrencies.

xxxxxxxx

9. Puzzles, contests, tournaments, prizes, adventures

   We do this through the XMOS Games, starting with Detroit kNights, the MMORPG game that is based upon Episode 4 of the book series, Cyber Robin and the Hoods.

10. A New MYTHOS for the World

    We do this starting with the games and tournaments.

More on OASIS System Framework and Taxonomy

OASIS as a complex ecosystem of software-based applications and special devices (most of which originate from other providers, such as phone, tablet, laptop, desktop and headset makers) is very similar to the core architecture of EcoVita. From another perspective, EcoVita is entirely embedded within OASIS. See § 4 below for more on the EcoVita architecture and software (as well as some of its hardware).

§ 4.1 Ontology and Semantics

OASIS is built from spaces which have relations with another another. Spaces are composed of processes which include different levels of activity and relationship. Some processes are relatively substantive in a 3-space sense and are "object"-like or ""thing"-like – thus, traditional objects such as geometrical forms. Other processes are not "things" to be experienced as 2d or 3d, but more about conditions, situations, and relational factors affecting how things in the spaces can behave – move, stand, change, and be sensed (e.g., seen).

These spaces with their processes are experienced through digital computational engines through scenes that are representations of spaces and their contents. Scenes are "views into" spaces. This is an important point and a distinction from standard "VR" thinking, including within the software that will be used for OASIS implementation.

The design of OASIS worlds and their contents is done principally through the use of scenes and the entities that are experience-able in those scenes. Operator-players in any OASIS activities are thinking and acting in terms of houses, tables, chairs, ponds, trees, avatars, and the actions that can be performed in a world of entities, some of which can be handled, changed, moved, etc.

Everything about spaces and processes is rather abstract. But at the level of scenes, we are now dealing with more familiar terms. Scenes have entities which are objects or other entities (more of the "relational" type) – such as cameras, lighting, and in the future, other factoring processes that can govern the experience, such as some sort of overall "dimness, brightness, fogginess, heaviness," (not to be thought of only in physical terms!) and other "general" phenomena that affect the scenes as a whole, including actions therein. See § 4.5 below for more.

There is a framework, a language of sorts, for spaces and processes, and it is called the SPF – Spatial Process Framework. Then there are other frameworks, for instance that of A-Frame, which will be used within OASIS for implementation of scenes and actions therein. There will be need for a way to move between different frameworks, and this will be provided by something known as the SPT – Spatial Process Translator. See § 4.5 below for more.

§ 4.2 The Twelve Rules of OASIS Phenomenology and Information Operations

Any object (entity) within OASIS can or does have these attributes or functions:

[1] Representation as a geometrical object – 2D, 3D, with animation and object-physics
[2] Any image or other media (e.g., video) on any face (surface)
[3] Link(s) to anywhere else, on any face, edge, point, or partial section
[4] Multiple agents resident in it
[5] Act as a processing node in a CHANT network
[6] Act as a database connected to some system including machine (e.g., sensor, actuator, robot) – following the principles of A,C, S, M – see EcoVita architecture
[7] Can move or be moved between OASIS worlds – with some constraints, rules, restrictions
[8] Be joined with other objects into a group
[9] Adhere to the Veblen axioms of projective geometry and sets
[10] Provide 1 or more sub-windows (other objects) of data (e.g., documents, graphics, charts, texts, videos)
[11] Be mappable to some real-world object/location
[12] Be secure, encryptable, hide-able, anonymous-capable, and totally private

§ 4.3 Decentralized Network Architecture of OASIS

The decentralization, load-balancing and fault tolerance is based strongly upon classical MIMD parallel distributed processing (CSP), and networking computing. Any device can be a computing resource, and most devices including mobile communication and IoT devices can be also servers (e.g., using mongoose from Cesana). Moreover, the server functions can be divided and distributed, and they do not need to all be on any given processor platform.

CHANT and ATHOS are important in this respect. Furthermore, OASIS will probably make use of technologies and products such as Enigma (www.enigma.co).

[also bring in material from other notes and docs]

We begin with the A-Frame platform developed by Mozilla because it offers a framework into which many different formats of representation and manipulation can be incorporated. In terms of the UX (user experience), everything can be operating through any compatible browser, as well as through explicit VR devices (e.g., headsets).

Following the A-Frame logic, there are entities and scenes composed of entities. Within entities there are objects that define geometrical shapes, plus there are cameras, lights, and other types of entities that are not explicitly 2d or 3d objects.

A scene provides for the representation of a space but here, within OASIS, we distinguish the representation from the underlying "ontological" framework. Actually, the fundamental thing of importance ontologically is the Relation between entities. The geometry, and thus the representation, and how it is experienced by a viewer-participant, may change. Initially, however, we are focused upon conventional Euclidean space and the conventional "everyday" experieince of being situated in and moving about in 3-SPACE.

That space will be a portion of some vek or (usually) a combination and assembly of veks constituting a room, building, neighborhood, city, etc.

Scenes are described in terms of their elementary geometry which by default is Euclidean (but in the future there can be worlds where there are strong variations leading to interesting Riemannian and Lobachevsky geometries!).

Scenes have entities in them, and these may be objects of different geometries, cameras, lights.

When the scene is composed, this is what the viewer experiences.

SPT – the Spatial Process Translator

The SPT operates by taking a space framework defined in one language and reformulating it in another, such as to go from SPF into A-Frame descriptions, or potentially, vice versa.

Much of what will be done in OASIS implementation will start from the languages used in various tools, such as A-Frame, and gradually some scenes with their object-entities will gain other attributes, features, and functions not usually found in VR-type environments.

Consider a scene composed of a room with various furniture. Avatars can enter and leave the room through doors. Everything can be defined as an <a-scene> through A-Frame, for instance. But initially, this is just a scene per se, nothing other than a description for what can be experienced on a computer screen or through a VR headset. The properties of objects – what map to and from basic SPF processes that have extension (dimensions in 2d or 3d) and other physical properties (color, texture, etc.) - are in scene definitions only such geometrical and physical-like properties as make sense for scenes, for visualization, navigation, and otherwise for experiencing by persons interacting with the scenes.

But there can be much more.

There can be informational attributes associated with any part of a scene, particularly with specific objects. Knowledge contents, for example, that can be accessed in particular ways.

There can also be interaction between a digital ("VR") space (scene) and something that is in the physical real-world ("RW"). This is where the AR and MR functions come into play.

A user can be experiencing things from either of two basic starting points:

RW
I see a cubical box on a table in a room. Everyone can agree that this is "really" the case because we can do measurements and repeatedly validate the results. But now, as I do certain actions (which may be implemented as settings and options for my UI (user interface) methods, I can see some kind of AR effects imposed, integrated, woven with the display-experience of the cubical box. Then I can interact in different ways with the box and its additional, augmented information contents. This may lead me into an entirely other scene, and it may entail a shift from one location in RW to another in VR.

VR
I see a cubical box on a table in a room. Everyone can agree that this is a "VR" box, table and room, because we can do different sorts of measurements and repeatedly validate the results – it is all "in the computer" and not "out there somewhere." But now, as I do certain actions (which may be implemented as settings and options for my UI (user interface) methods, I can see some kind of MR effects imposed, integrated, woven with the display-experience of the cubical box. This may be a "tunnel view" into something that is "RW." Then I can interact in different ways with the box and its additional, mixed/augmented information contents. This may lead me into an entirely other scene, and it may entail a shift from one location in VR to another in VR.

§ 4.6 Spaces, Processes, Activities, Functions, Modules

This is a further refinement that supersedes anything different both above and below in this workbook or any other document up through the present (10.October.2018).

Refer to the diagrams in oasis-intro-overview-workbook01.ppt for more clarity. Here are four key ones, but note that there are some changes in terminology from the texts in these diagrams.

A Summary of Fundamental Terms

OASIS is composed of worlds, as described elsewhere and earlier here. That is all in terms of topologies, structures, movements, objects.

Another way is the following, and this is more about the logic of operations.

OASIS is composed of Spaces wherein there are Processes.
The Processes all involve, from the perspective of interactive user-participants, Activities (formerly referred to throughout as "Primary Functions" – but we want to clarify the use of "Function" within all this).

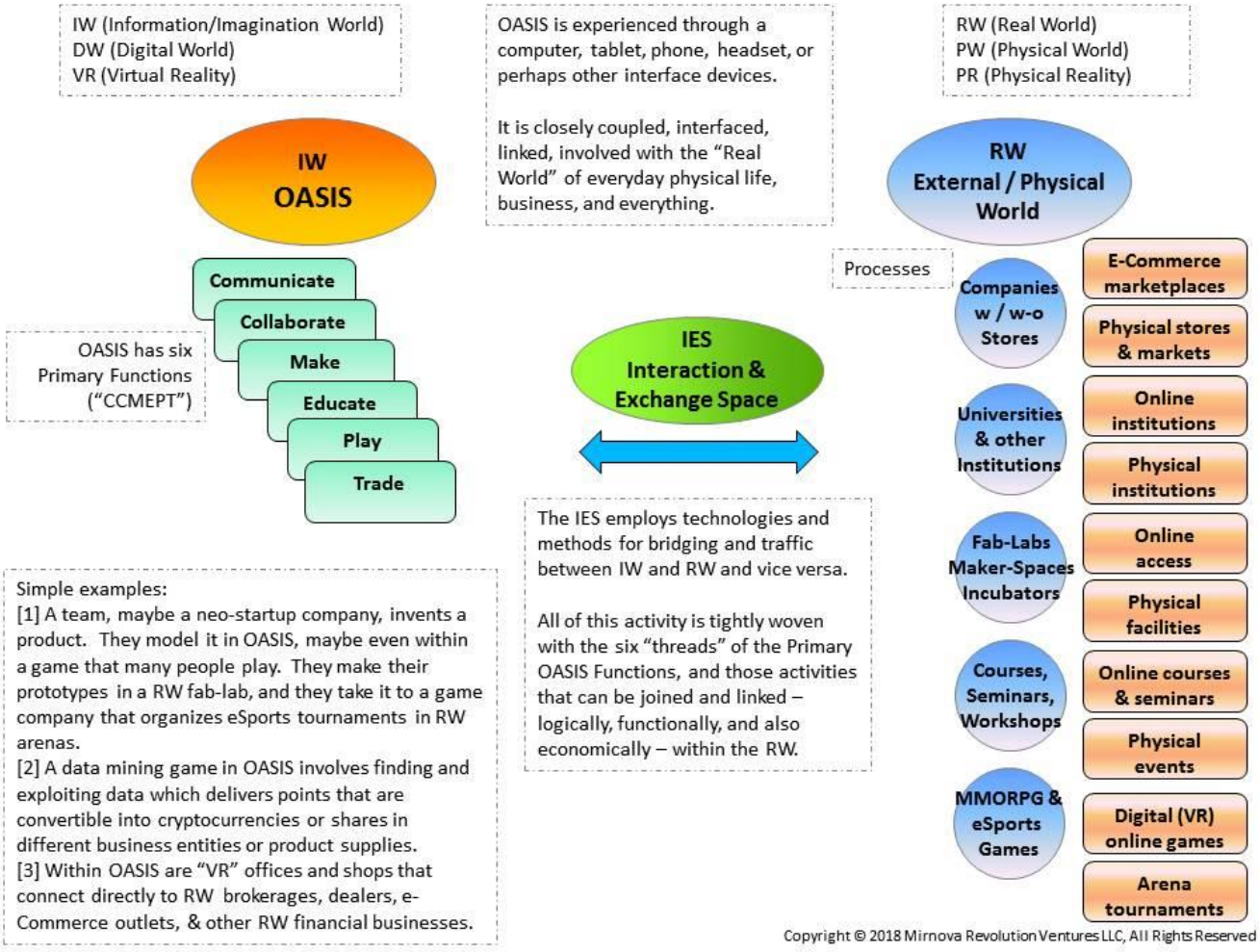
Activities are: Communicate, Collaborate, Make, Educate, Play, Trade ("CCMEPT", pronounced, "Smart").

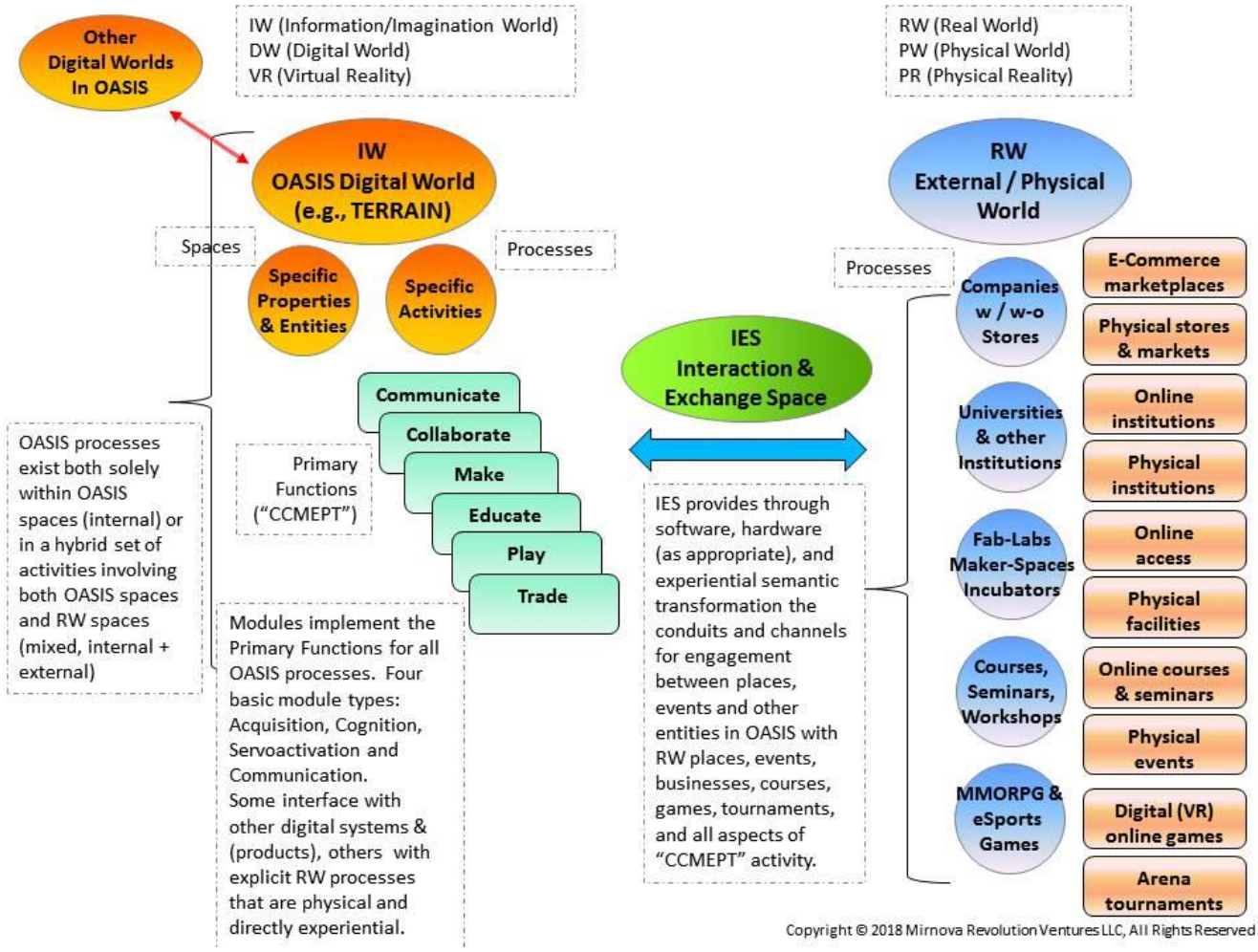
Processes are composed and built up of Functions.

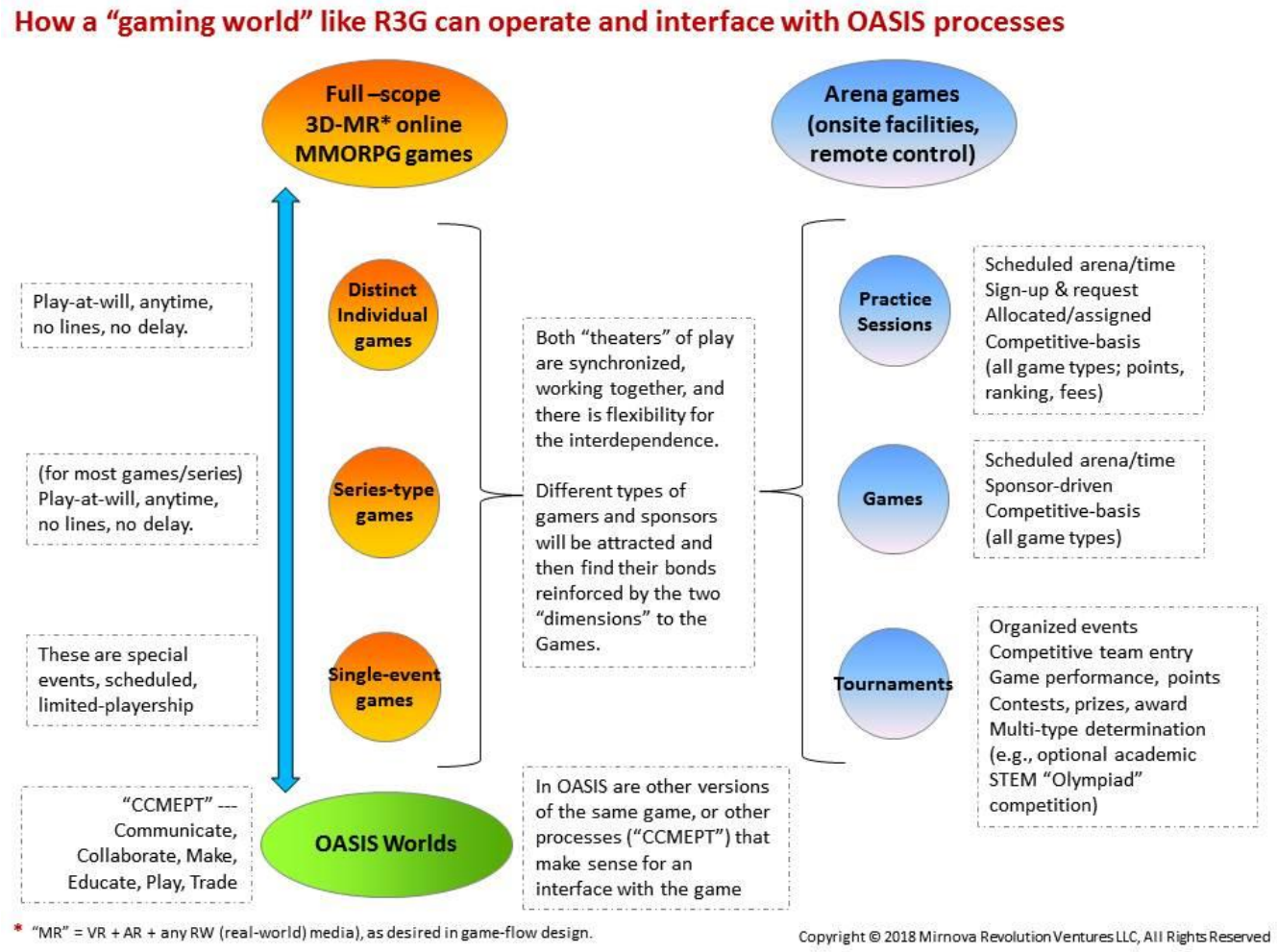
Functions are implemented through Modules. Functions are ultimately in one of four basic alsses: Acquisition, Cognition, Servoactuation, and Communication (A, C, S, M).

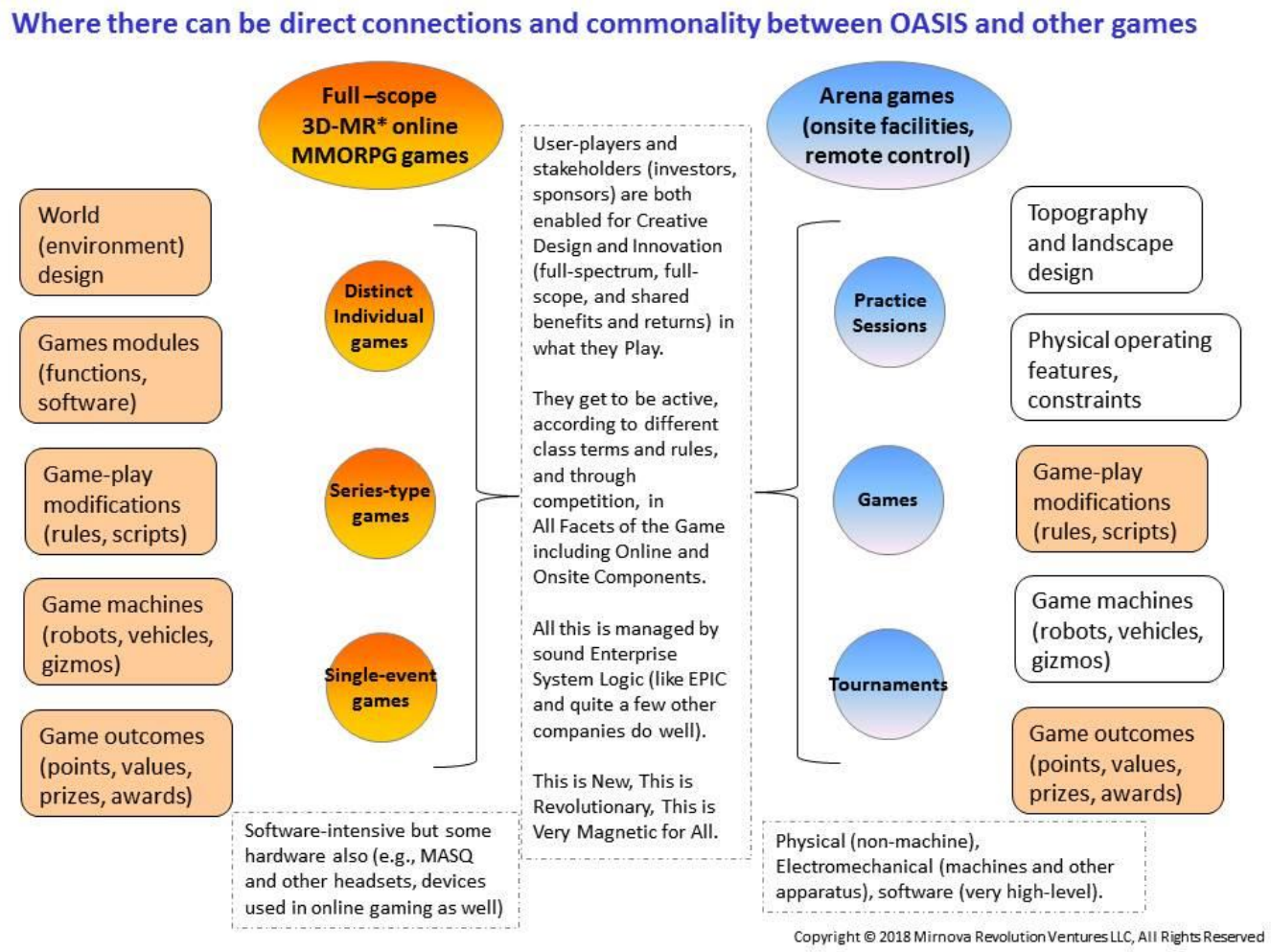
Modules are purely logical but ultimately involve software and hardware.

These figures need updating, MAINLY IN TERMS OF REPLACEMENT OF THE TERM, “Primary Function” with “Activity”.









[end of new edits made on 10.October, @ 2pm]

§ 6. OASIS Principal Worlds and Cities – TERRAIN and Others

OASIS has one unique and special World which is also the first and it is created by MREV and it remains under the principal ownership of MREV.

This is known as TERRAIN, also as T'Rain. It is a spatially-temporally enhanced and modifed Earth.

§ 6.0.1 Four Primary Worlds

There are actually four (4) Special (Primary) Worlds:

TERRAIN (T'Rain, TERRA)

LEO-1 – Lagrange Earth Orbital One – a MOSES-type network-cluster space station

LUNA – a base on the Moon

ASTRIC-Net1 – a space station beyond lunar orbit and focused upon ASTRIC operations for asteroid deflection, deterrence, and also mining and industrial uses.

§ 6.0.2 Sixteen Special Cities within T'Rain

TERRA (T'Rain) has multiple developed urban and mega-urban Special Cities
There are sixteen (16) such Special Cities. They are variously connected in different degrees to present spatiotemporal locations, histories and events.

Five are particularly linked with the Gorskiwan Epics and the stories that connect also in different ways with Ordo Lucis.

[1] Alpha-Skiv – Scythian “Amazon” city complex in SE Crimea
Modern-contemporary with roots to ancient times and Skiv civilization. Njuilt into seaside cliffs, mountains, ravines, valleys
Female-dominant-governed society
(Earth)

[2] Podmoskva – Undergroun-mainly super-tech, the ultimate EcoVita Complex, situated in and under Moscow
Ultra-modern but contemporary
Mostly egalitarian, M/F-balanced in governance as a society
(Earth)

[3] TeraNod
KZ-Semipalatinsk region)
(Fire)

[4] ArcoSolaris

[5] Gorskiwanao
Siberia
(Earth)

[6] Lyamzia-Capital
(Air)

[7] Perico
(Earth)

[8] Atalan
(Water)

[9] Croloman
(Earth)

[10] Templar Cross
(Earth)

[11] Noveau Traverse Cité
(Earth+Water)

[12] New TokiOsaka
(Earth+Water)

[13] Siamagon
(Earth)

[14] Amazonas (Manaus)
(Earth+Air)

[15] (a Viking name)
(Earth+Fire)

[16] Anatarcticus
(Earth + Water)

§ 8. Progression to OASIS System Foundations leading from EcoVita and AgroIntel

What is in EcoVita, and specifically initiated for the initial, core module, AgroIntel, provides the ready-to-shift-and-reshape foundation for what goes into OASIS.

Thus, this material is presented here with very little change or commentary. All that is for later in the maturation of the system architecture development.

Synopsis of the EcoVita-based technology:
EcoVita = a super-system or system-environment with one basic architecture and three main components.
EVA = abbreviation for EcoVita
EVA = an Intelligent Cybernetic Environment (”ICE”) – it is a set of systems for performing adaptive intelligent control and optimization. These components are as follows:

- AgroIntel = the modular system of hardware and software for control networks in Agriculture
- IntelErgy = a comparable system for Energy
- IntelEco = a comparable system for Environment (ecosystems)

There are a few related important terms:
AgroBrains = the educational component (educating, training and mentoring) for youth and adults in the

principles of “smart farming” and intelligent agronomics.

S-Water = Smart-Water, a specific module within AgroIntel that is the basic introductory module.

But in terms of OASIS, replace those components with Worlds and Games and Modules for the Six Core Function Areas.

~~~~~~

EcoVita (“EVA”) is an open-ended extensible platform for measurement, control, and prediction functions pertaining to agriculture, environment and energy operations and management. It is a system for performing both physical tasks and building a knowledge resource that can be used by the operators of its networks and by others who require the information produced within EVA. It is a “system of systems” in that it consists of three component systems: AgroIntel, IntelErgy and IntelEco, of which the first, AgroIntel is the main focus at present (2018+) and the core system that defines the architecture and design principles and implementations that will be used in the other future component systems.

EVA is thus described as an Intelligent Cybernetic Environment or **ICE**. The main objectives are to provide human+machine functions (including semi-autonomous and autonomous operations) that will enable adaptive intelligence in the control and optimization of agricultural, energy, and environmental management tasks.

The ICE is exactly what we want within OASIS.

Initially, EVA is designed to serve principally rural and reduced-infrastructure farming and residential communities. This includes agriculture operations and energy management within regions and locales that for one or more reasons are considered to be difficult, harsh, inhospitable or extreme for traditional agriculture and/or energy production. EVA is also intended to be employed within education involving “applied STEM” including electronics, mechanics, and software engineering, including multiple types of robotics, signal processing, machine learning and other functions. Such educational activities are conducted at multiple levels of educational progress and competence, with elementary components serving youth in middle-school and high-school levels, adults in university and post-graduate levels, and adults pursuing continuing education. This activity has an initial and dominant focus upon agronomics and agriculture and is known as AgroBrains.

AgroBrains is focused upon the use of AgroIntel modules and other related and compatible technologies and products including UAV and UGC robotics, space agriculture informatics, and basic agronomy. AgroBrains constitutes activity, organized and coordinated by MIRNOVA with partner institutions and people, that teaches and facilitates the use of AgroIntel and related technologies for improving agriculture on operating farms, improving job skills, and for co-facilitating employment and agri-product commercialization including farm produce marketing and sales.

EcoVita employs principals of modular and object-oriented design, hardware and software platform independence, device independence and interoperability, asynchronous parallel processing, network computing, and other architectural features that enable it to be expandable from single-function modules to very large arrays and networks, for instance serving large commercial farms or energy generation grids.

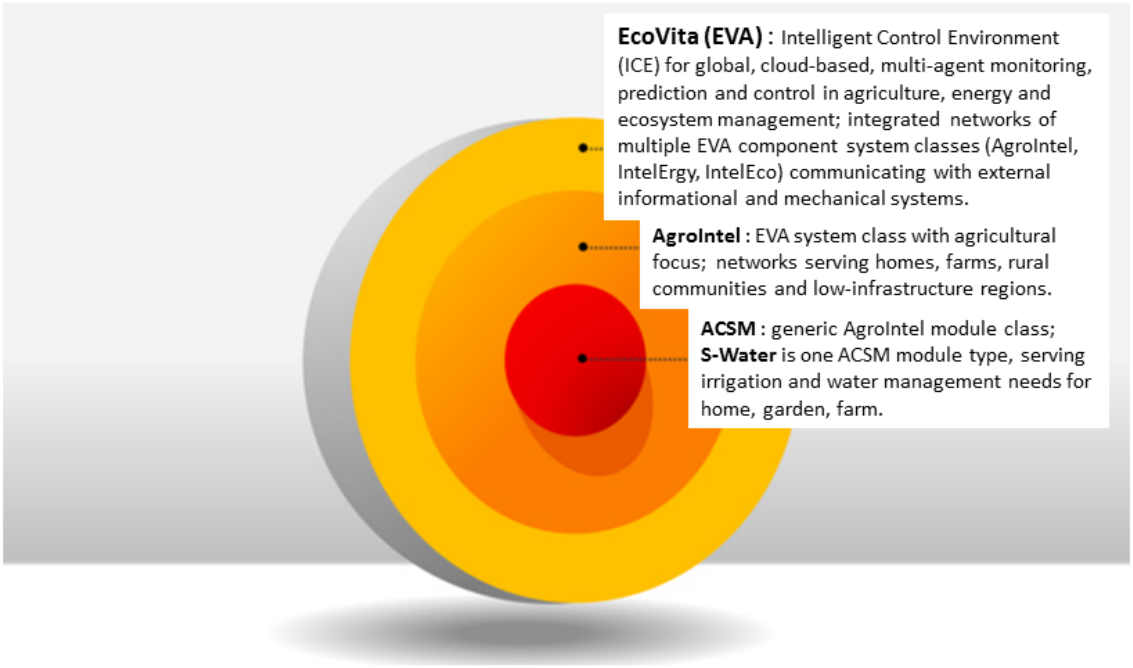
~~~~~~

EcoVita is being designed and constructed in an expanding shell of functions and services, beginning with certain very basic agricultural-focus modules serving needs in seed crop planning, irrigation, fertilization pest control, pruning and harvesting.

Within AgroIntel the first module is S-Water ("Smart Water") and it is an example of a generic ACAM module within all of AgroIntel and future EcoVita modules and networks. "ACSM" stands for "Acquisition-Cognition-Servoactivation-Communication" as these are the four elementary module functions. We can think of S-Water as the primary generic EVA module but one that also begins its life as a provider of irrigation functions in AgroIntel usage.

All of this – all the preceding remarks – pertain to OASIS. This includes the "Brains" components, focused upon Education and Training. It's just that now we are not talking about specific electromechanical functions on farms, or in rural energy grids, or in environmental management, but about a wider range of functionality pertaining to virtual-world (VR) and real-world (RW) communities, lifestyles, games, economies.

Figure 1 illustrates the high-level architectural model that governs design and implementation.



EcoVita (EVA) is a class-based architecture for networks that provide informational and mechanical control functions using modules that contain four functional submodules: Acquisition, Cognition, Servoactivation, Communication. Each module instance has specific tasks, some that are network-shared, reconfigurable and reassignable among other modules.

In addition to its assigned specific class and instance tasks, each module can serve in a MIMD parallel processing network (CHANT, Banyan CSP) providing multi-directional data throughput and segmented-task processing using BOINC-type distributed network computing with load balancing across the local network and potentially the global EVA network.

EVA operates, as a background set of computational tasks, the data mining, analysis, pattern recognition, and learning that constitutes the SELDON Prediction Engine (forecasting environmental, climate-related, energy, agriculture outcomes).

Figure 1 – EcoVita system model

One can extrapolate here very easily from EcoVita-explicit to OASIS-implicit.

§ 8.2. CHANT (Banyan)

This is straightforward at the heart of OASIS system environment. It should not be hard for the reader to see the transitions and implications.

The system is based upon a functional network model named CHANT (also known as Banyan). The system consists of independent, asynchronously communicating nodes which are integrated together in a modular, expandable, heterogeneous network. The Banyan Model incorporates the ATHOS multi-agent computing environment ("ATHOS" = Adaptive set-Theoretic Hierarchical Operating System), which is a semantic logic based upon communicating sequential processing (CSP) derived from process algebras (Hoare, May, Milner, Hey, Ericsson-Zenith, et al).

AgroIntel is designed to work on multiple scales of complexity with multiple user audiences (communities), irrespective of socioeconomic status or background, formal education or training. It provides both semi-autonomous and autonomous control for an open-ended set of functions. Programming and network management may be done using any one of several user interface schemas which enable working with different application toolkits, ranging from an HTML/XML-compatible scripting language to programming in one or more conventional interpretive and compiler-based languages (e.g., Python, Lisp, Scala, C++), to a simplified graphical programming tool based upon the parallel languages of OCCAM and Linda.

§ 8.3 CHANT Module Processes and Classes

We begin by discussing module super-classes, then basic module processes, then submodule types, then different classes of modules.

§ 8.4 CHANT Module Super-Classes (Fundamental Types)

EVA modules consist of <u>two fundamental types or super-classes</u>:

Module Super-Class: **L-type** **M-type**

<u>L-type</u> (logic, lambda, **λ**) = software-only; modules of this type are loaded onto other modules, on which they are executed by the computing processors of those modules. L-type modules perform processing that results in some further actions being conducted by one or more other modules of either L-type, M-type, or both.

<u>M-type</u> (machine, mu, **μ**) = electro-mechanical; modules of this type are connected either with physical links or by wireless connections. M-type modules typically have microprocessors and other digital electronics and differ from L-type modules by virtue of including some type of non-digital

electromechanical apparatus. (An M-type module could, in theory, have no computing processor and no software, but this would be a rare exception, and from the perspective of design, such a module's software would be the NULL operator.)

Each OASIS world-system is composed of an open-ended and extensible network of modules (software and/or hardware; L-type or M-type) that include members of several module classes, any of which may be incorporated into any of the above three subsystems (AgroIntel, IntelErgy, IntelEco) with minimal adaptation and adjustment of either hardware or software. Modules are abstract machines that are composed of physical (hardware) and/or informational (software) components and usually both. (Different modules are presented and described below.) The distinctions among modules pertain to specific electronics hardware, physical apparatus, and software that together constitute the module architecture definition.

In principle, any CHANT module can be physically and informationally integrated into any instance (case) of any type of CHANT system. Integration between modules involves the following generic processes, depending upon whether software or hardware or both are involved; these are discussed further below.

§ 8.5 CHANT Module Processes

Every CHANT module performs four basic logical processes.

Module Processes: **Acquisition** **Cognition** **Servoactivation** **Communication**

Note: This applies to much more than EcoVita (EVA) which is focused upon many physical activities involving physical sensors, actuators, processors. This applies as well to every VR process in OASIS. It is important to read these points into all of the text in this Section 8 of this workbook. Where one sees," EVA," understand that it is CHANT and it is also for everything in OASIS.

Acquisition - Sensing, detection, data acquisition from a source that is either internal to the module ("built-in") or external to the module (in some other module or external to the entire CHANT system). Acquisition is distinct from communication, another basic process. Acquisition is active – programmable, changeable, responsive to situations and conditions internal to the module (other co-resident, co-operative processes) or external within other modules including external systems not directly within the CHANT system network but with which it is linked and communicating.

Cognition - Reasoning operations – calculations performed on the data streams within the module that are generally more complex than simple arithmetic-logical functions or procedures. These may be simple or complex algorithms constituting synthetic ("artificial") intelligence operations, including use of rule-based and neural-network models, with or without human interaction and decision-making. These are the processes of evaluating singular or multiple data streams, generally real-time, possibly accessing databases and statistics, possibly using information obtained from previous operational histories, from satellite telemetry and/or robots such as UAV drones.

Servoactivation - Activation, servo-control – the command-and-control functions pertinent to operating electromechanical devices either internal to the module or external, similar to Acquisition processes.

Communication - Receive/transmit operations – programmable, controllable, active receiving and sending data to other modules or to an external, non-EVA system. This is more than simply "read/write" or "push/pull"

sharing of data. This involves some type of processing that changes data structure and is dynamically responsive to conditions and signals within the module and coming from other modules in the network.

By default, all modules will perform actual (software and/or mechanical) communication and cognition processes, since the module functions as part of a network. If a given module does not have specific tasks to do in acquisition or servo processes, then those processes are implemented as (Null, No-Op) functions (in the conventional software sense of those terms).

§ 8.6 Module Submodules

All processes are executed (conducted, performed) through the use of submodules which, as with the modules themselves, are either purely software, purely hardware, or both. Typically a given module will have one submodule for each process type. However, there may be multiple submodules for a given process within a given module.

Module Submodules: **Acquisition Cognition Servoactivation Communication**
An EVA module will have, in principle, 1 or more submodules (software and/or hardware) for each of these basic process types:

Acquisition submodule (acq_Submod)
This contains an acquisition sensor-set (1 or more devices that acquire data – note that for various reasons in a module design or operation these may be NULL or No-Op devices, but logically, there is at least one). Thus, an acq_Submod may have multiple sensor devices feeding into it. There can be multiple acquisition submodules in a given module, each of which can have multiple sensor elements. (Example: In the initial, elementary S-Water module, there is only one such device, a hydration measurement sensor.)

Cognition submodule (cog_Submod)
There is at least one such submodule in a given module. However, it may have a logic-set consisting of many cognitive units (elements), and these may be both software and hardware. (Initially, within EVA design and development, they will all be virtual cognitive elements that are implemented in software (e.g., executing on the Arduino microcontroller core).) (Example: In the initial, elementary S-Water module, there is only one such process which decides on the basis of sensor measurements and external data, how to control the pump and manage the power and water supplies.)

Servoactivation submodule (srv_Submod)
One or more servoactivator-sets which connect to devices, virtual or physical. Depending upon the module function and design, there may be multiple devices that are controlled by the given module. (Example: In the initial, elementary S-Water module, there is only one such device, a water pump.)

Communication submodule (com_Submod)
One or more communication-channels, which connect to other modules. These channels provide for bi-directional transmission of data that may be 1:1 (point-to-point, destined for the recipient module only), or 1:n (many; serving to supply multiple modules with the data). For a given module, the data transmission may be intended for that module or it may be purely something to be passed on to others elsewhere in the network. (Example: In the initial, elementary S-Water module, this submodule communicates data with other modules and with external user application(s).)

Figure 2 illustrates the general logic of EVA modules.

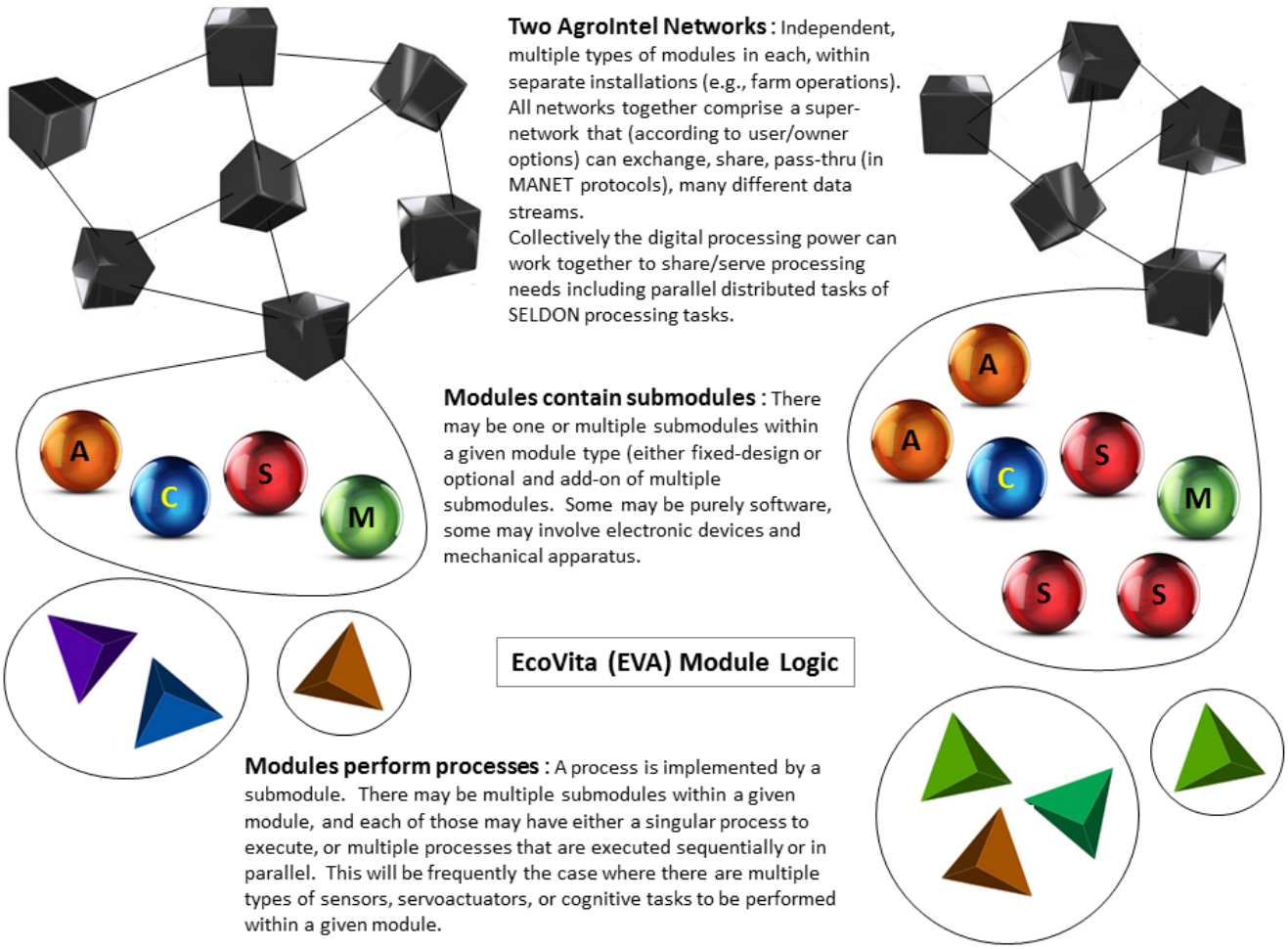


Figure 2 – Generic EVA network module, submodule and process architecture

§ 8.7 ACSM Module – the Generic EVA Module Class

What constitutes the generic module in any EVA system, operating as a CHANT (Banyan CSP) network, including AgroIntel? How is this implemented as an ACSM module (ACSM = acquisition + cognition + servoactivation + communication), such as S-Water ("Smart Water")? What functions do these and other specialized modules perform? How do the modules communicate, share data and information, produce knowledge, and create results that include the physical control of various apparatus such as pumps, sensors, solar panels, wind generators, and robots? We examine the foundations of the basic modules in any EVA system, focusing upon AgroIntel.

A generic ACSM module has all four submodule components, corresponding to the four distinct types of process as described earlier (§ 2.2 and 2.3):

Acquisition Cognition Servoactivation Communication

There is at least one logical component for each process. That component may be a “no-op” (“null”) in terms of physical action and data manipulation, but it is still a logical element of the module.

Some data is acquired from outside the module through an *acquisition submodule*. This process is called *sensing*. That data is processed and constructed into information within the module, through a *cognition submodule*. This process is called *reasoning*. Some information is used to generate data that is submitted (as command-and-control) to a *servoactivation submodule*. This process is called *servo* or *activating*. Some data is communicated to one or more external modules through a *communication submodule*. This component also handles communications from other modules which occur asynchronously. This process is called *transmitting*.

This is a simplistic description, because, in even the most basic generic module, there is communication going on with potentially n other modules, incoming and outgoing messaging, there is feedback signaling from the action submodule, meta-data from all submodules, and a variety of internal processes within the cognition submodule. However, we should keep in mind that there may be modules that are more specialized and these will typically have only 1, 2, or 3 of the basic submodule components. The most generic is also the “universal” module, and it is known as S-Water (“Smart Water”).

§ 8.8 Modularity and Expansion

AgroIntel and other EVA systems are completely modular and expandable in all functional and dimensional parameters. The underlying principal here is comprehensive interoperability, platform independence and “universalizing” configurability:

Any type of EVA module may be introduced into an EVA system without software modification and reprogramming or other external (human or agent-based) adjustments to the software of any existing module within that system.

Any type of EVA module may be introduced into an EVA system without hardware modification and re-engineering or other external (human or agent-based) adjustments other than the nominal actions of physically connecting appropriate and necessary subcomponents (e.g., wires, hoses, moving mechanical parts).

§ 8.9 Module Parameters and Function Sets

There are defined sets of functional parameters for each module class. Within AgroIntel, for instance, there are modules dedicated to irrigation, pest control, fertilization, crop inspection, predator monitoring. Each module type has a functional parameter set linked with a particular class and instances of some device (e.g., sensor, actuator, pump, sprayer, robot) and its control software. This module-centric functional parameter set is known as the **f_Set** and for a given module class, there is the **f_Set[m]** where “m” denotes the module.

Within each f_Set are parameters that pertain to specific submodules within that module. These are denoted by the following high-level schema of representation and denotation:

f_Set[m] = the set of all the parameters used by a given module

f_Set[m].[submodule-identifier] = the set of those parameters within f_Set[m] that pertain to a given unique submodule. (Bear in mind, there may be more than one instance of a given submodule type for a given module.)

The aggregate of all functional apparatus within an AgroIntel system is managed through the processing of

functions that collectively operate upon the aggregate of all modular f_Sets, f_Set[m, s.t. 1..m..n], and that aggregate constitutes the system function-set (denoted by the term, **F_Set**).

§ 8.9.1 Parameter Types

Parameters are of two types: operational and dimensional.

Operational parameters are of many varied types and these refer to actions performed in the processing of acquisition, cognition, servoactivation and communication. Operational parameters are "general, typical, usual" when one thinks of actions being done by the modules.

The operational parameters employed by a given module comprise a subset of the module's f_Set[m].

Dimensional parameters are of two subtypes – spatial and temporal.

- Spatial parameters define the physical spread (distribution, range) of the system (e.g., covering an expanse of several hectares that constitutes one farm).
- Temporal parameters define how the system will be employed in terms of time intervals and duration (e.g., every day or every other day or once per week for the next six months).

The dimensional parameters employed by a given module comprise two subsets of the module's f_Set[m].

§ 8.9.2 Module-level Parameter Representation

Thus, there is this schema of parameter representation and denotation:

System level: F_Set (for a given EcoVita system instantiation, such as one AgroIntel system.
Module level within a system: f_Set[m]
Submodule level within a module: f_Set[m].[submodule-identifier] or f_Set[m].[sub]

Within any given f_Set[m].[sub] there are:

f_Set[m].[sub].s --- spatial parameters
f_Set[m].[sub].t --- temporal parameters
+
f_Set[m].[sub].a --- operational ("a" for "action") parameters

§ 8.10 General Module Classes (Types)

These are the elementary classes of EVA modules. All specific modules (e.g., S-Water) belong to one of these fundamental classes. (Note: these are not all the "permutations" of the four process types). These are also referred to as function-types (f_Type) for the modules.

Primary ("Simplex") Types

These have one submodule only for any process; thus, a maximum of four submodules in a module, and no module has more than one type of submodule for a given process type. Recall that "Communication" as a process denotes operations over and beyond the simple "read/write" or "push/pull" of data from resident memory in a module.

The most common:
ACSM – the most generic and universal, including one submodule each for the four process types (acquisition, cognition, servo(activation), communication). Within AgroIntel, the initial module, S-Water, is designed as an ACSM module.

AM – acquire and communicate; no "AI" cognitive functions to perform, nothing to be physically controlled.

ASM – acquire, perform some servoactivator functions, and then communications.

CSM – receive network-internal information, perform cognitive tasks, control some servoactivator(s), and communicate results to other nodes (modules).

Less common:
ACM – acquire, perform cognitive functions, and communicate; no servo functions to control.

AS – acquire data and perform some servoactivator process.

AM – acquire and calculate what communications need to be performed and execute those accordingly.

S – fairly simple and straightforward servoactivation under fixed commands and reporting.

M – basically performing a "bridge" function to handle (schedule, coordinate) multiple communication channels.

Advanced ("Complex") Types
These are the same basic module classes as above in the Simplex type, but where there are multiples of one or more of the process types (and thus, submodules) within a given module. The basic functionality is the same as for the Simplex types.

§ 8.11 AgroIntel Network Growth (expansion of modules)

A user may begin with one single module, such as the S-Water Module (described further below in this document) which manages irrigation using one humidity sensor, one pump, one electric power supply unit, and one distribution unit.

This elementary module is programmable at several levels of variety and complexity in functionality. Additional modules may be added by the user with a minimal of technical challenge and with no mandatory required changes to hardware or software. Expansion of the network from one to n modules is characterized by the following attributes:

- Add 1 to (n -1) modules of any class – they may be all the same type (e.g., the S-Water module – described further below here) or a totally heterogeneous set of modules)
- Expand the network by increasing the spatial or temporal operating parameters and complexity of operation without changing the basic F_Set structure of the initial network state. Thus, the network gets deployed to larger areas of land, for instance, or its operations are extended to longer hours of

each day, week, month, etc.

- Expand the network by increasing the functional space without changing the spatial or temporal operational state-space. This is potentially distinct and different from the first point made above, regarding heterogeneous modules. There are potentially multiple functions that can be performed by any given class of module or any given subset of modules (e.g., m1, m2, m3, … m(i), where m[i] denotes a unique module instance. However, this is a matter for more advanced system implementations.
- Expand the network by modifying (together) the functional, spatial, and temporal parameters. Thus, the system grows from something like a home use to a small farm to an agri-business operation, with hardly any changes required to any hardware or software modules (!) in the process, other than simply, literally, adding units, positioning them and in some cases physically connecting them (although almost everything is and will be wirelessly linked), and leaving everything else up to the control software(!). This is a remarkable feature of the ATHOS operating logic paradigm and the Banyan CSP model. The software accommodates additions, subtractions, and (in terms of subsets of units) multiplication and division. One can say that the network "does its own arithmetic."
- Is there more that can be done with this ultra-simplistic architecture? Certainly.

§ 8.17 EVA Module Relational States

Any module may interact in any of several different and dynamic functional capacities (roles) as a component within its network processing environment. These roles are known as Relational States.

Relational States: **Alpha Beta Gamma**

Alpha (control and coordination of 1 or more other modules – "supervisor")
Beta (performing tasks under the supervision and control of an Alpha module – "worker")
Gamma (performing connective and communicative tasks between 2 or more Alpha-Beta module sets – "messenger"

Since EVA modules are all functioning in the context of real-time parallel processing, any given module may operate in a given relational state relative to other modules and the processes being executed on those modules. This applies to any given system context and at any given time. This will be utilized for more complex instances of EVA as development proceeds for larger and larger NPE and applications.

Some modules can be assigned (permitted, mandated) or de-assigned (limited, prevented) the ability to be in one or another relational state. This will depend in part upon the module type, but also this may be dynamic, and connected with performance of the overall system network.

§ 9. CHANT ("Banyan") Architecture

The overall system architecture of OASIS is known formally as:

Cooperative Heterogeneous Asynchronous Network Transprocess (CHANT)

CHANT is a model for computational and mechanical processing that involves the cooperative actions of devices

which are of heterogeneous architectural types and methods of operation and which act as asynchronous and independent nodes within an amorphous and non-deterministic network, communicating with one another and sharing both data and program tasks, performing processes that are distributed across (trans) logical and spatio-temporal subsets of the network.

CHANT is based upon a deep foundation of computer science and information technology. This includes lambda calculus, process algebra, CSP (communicating sequential processing, itself based upon process algebras), PDP (parallel distributed processing), MIMD (multiple instruction multiple data parallelism), OCCAM (parallel processing in a MIMD CSP/PDP environment), multi-threading and multi-core processing, multi-agent paradigms, and different forms of machine learning and synthetic intelligence.

This architecture is also colloquially known as Banyan, derived from the metaphor of the banyan tree. CHANT and Banyan are synonymous; the former is a technical term, the latter is a commercial product name.

Within the modules comprising a CHANT system (e.g., one OASIA world-system (or an EVA system, such as one instantiation of an AgroIntel class subsystem serving a farming enterprise), there is a shared, distributed, network-pervasive meta-control framework for managing information-sharing and knowledge resources. This framework consists of software that performs tasks designed to discover and learn knowledge from the information traffic that exists in the processing by all the modules.

§ 10. ATHOS (meta) Operating System

ATHOS (Adaptive set-Theoretic Hierarchical Operating System)

ATHOS is not the same as a traditional, conventional “operating system” (e.g., UNIX, Linux, Windows, MacOS, Android). Such other “OS” software is employed, according to the specific module requirements, for all the usual data and program management tasks. ATHOS is a software system whose purpose is the control of information flow for the purposes of machine learning and knowledge generation. It operates as a background set of computational processes, distributed across the entire CHANT network (e.g.,), and it makes use of all resources on all modules (and thereby, all processes of acquisition, cognition, servoactivation and communication).

ATHOS is the engine for knowledge engineering and adaptive synthetic intelligence (ASI) that serves two major purposes within EVA:

- Control and optimization for different devices in present-tense real-time operations
- Prediction and forecast for future operations and/or analysis, planning and decision-making external to EVA-specific operations

The control functions produce data both directly within EVA modules (e.g., measurements from sensors, power units, flow meters, image analysis, etc.) and indirectly (by acquisition from other non-EVA systems including a large and open-ended suite of satellite and ground data, some of which is obtained by EVA for its control functions, and some of which is “pass-through” and simply accessible in the matter of course of operations. The latter is useful in EVA predictive functions.

The predictive functions employ multiple types of statistical and network-based algorithms, including

- Randomized stochastic perturbation and approximation (SPSA)
- Cellular automata
- Neural networks
- Genetic algorithms
- Bayesian networks
- Rule-based logic programming

In order to create forecasts of environmental conditions including extreme climate behaviors, energy utilization expectations, irrigation and fertilization requirements, seed planting, pruning, livestock maintenance, harvest planning, and crop routing to different distributors and markets. Ultimately these predictive functions, through learning models, will provide a capability for a wide range of Futures Forecasts including market prices for various commodities (crops produced and supplies required).

§ 11. SELDON Prediction Engine

The Prediction Engine within EVA, known also by the name, SELDON,[1] is really the "deep-down, long-term" main objective of building and deploying the EVA systems in the manner by which they have been and are being designed. This is complementary, in parallel, in support by and in support for everything else that EcoVita serves. In other words, a major purpose and mission of EVA is to enable the implementation and deployment of a very powerful engine for knowledge generation, inference, and prediction, using agricultural, energy, and environmental information that has been acquired from a massive and open-ended variety of data sources including all of the acquirable performance data (virtually everything that happens) of each and every EVA system, and external data such as satellite telemetry and other agricultural, botanical, meteorological, geographic, and socioeconomic data.

Everything that comes "close" to any AgroIntel system implementation or that happens within its network will be collected and absorbed by the appropriate modules and reported to the SELDON Engine. The latter operates as a fully parallel, distributed, "cloud-based" server network that follows BOINC principles and is implemented across all available EVA nodes, in all networks, all system types (e.g., AgroIntel, IntelEco, IntelErgy).

Functionally, the massive data collection is not dissimilar to what has become in vogue as a normal practice by many internet service providers of search engines, social networks, and services such as email, video, chat, and file-exchange. Thus, one may think of many such providers: Google, Facebook, Instagram, Pinterest and Twitter, to name some of the "majors." However, there are dramatically significant and important differences. SELDON is not collecting personal information nor any data that could cause concerns over privacy, propriety or proprietary interests. SELDON is amassing public and voluntary data, and information produced within EVA, and knowledge generated within EVA, that is about agriculture, energy, and environment conditions, situations, practices and outcomes that go directly back in sensible, usable results to all of the EVA users – individuals, companies, public agencies, and other entities – and all of this serves to improve, to fine-tune, to continually optimize and make "smarter" the functioning of all the EVA processing within all the networks.

[1] "Seldon" refers to the central protagonist, Professor Hari Seldon, in Isaac Asimov's famous "Foundation" trilogy (later expanded) of novels from @ 1950.

SELDON is a purposive Knowledge Machine and Prediction Engine. It is not a search engine, nor is it simply providing massive estimates and conclusions that can be used by advertisers, particularly those that work by directing website visitors to particular target ads and such materials. SELDON can be employed to answer questions about matters concerning agriculture, energy and environment whose solutions may depend upon massive data deriving from behaviors of many people and intelligent systems, such as those implemented within EVA, operating over lengthy periods of time.

SELDON employs search engines, including any and all that may be available to it, and to the extent that its algorithms will deem it necessary, also other applications and data sources such as may be publicly available and generally of an open-source nature.

In the future, yes, SELDON can be directed toward other types of forecasting and prediction. However, initially, we begin with food and energy and what people use and do in order to generate their food and energy. That is enough for now. As the Prediction Engine operates, it naturally matures, refining itself – it has a self-learning capability. Gradually, it has exposure to more diverse information, and to more mistakes, and to more corrections. The best strategy for developing such a Prediction Engine that can become increasingly more generalized in scope, is to begin with a strong concrete focus upon knowledge problems that involve very concrete, tangible data spaces, and very concrete, tangible prediction targets. Soil conditions, humidity, fertility, cultivation success and failure, and final yields in terms of crops – all of this makes for the ideal experimental workspace for developing a Prediction Engine that can progressively undertake other tasks beyond the confines of the field, greenhouse, pond and pasture.

(More information on SELDON architecture and design is available in other documents, particularly those by MJD from 2016-2017.)

§ 12. More on CHANT Modules

§ 12.1 Module Classes

EVA modules exist in different functional classes, according to the superclass in which they belong. These classes define what types of submodule will be employed within a given module. Each module has a functional type (known as the f_type).

Refresher: Module Super-Classes are of two types: L-type (logic, lambda, **λ**) = software-only, and M-type (machine, mu, **μ**) = electro-mechanical.

Any class may have a hierarchy of classes that are members of that class, inheriting various attributes and behaviors. A class may also derive from multiple "parent" classes – there is not a strict "classical" object-oriented design.

§ 12.1.1 Mirror Types

Note that for any given M-type module (i.e., a module with some "machine" functions other than digital (or analog) electronics for computation), there exists a "mirror" module that is an L-type. The latter can be used in place of the M-type module for any purposes of simulation or in certain cases for inclusion within an operating

EVA network deployment. The L-type module provides what can be termed the “virtual module” capability that is intended to be extensible to any module within a network.

§ 12.2 Module Function Types

Each function-type (“f_type”) distinguishes the class from all others. Each class will have a unique configuration (set) of submodules. Whether as software-only or with hardware, each submodule comprises the process for some definable, finite, concrete tasks. The f_type is linked with what the module can do, with its different submodules (acq_Submod, cog_Submod, srv_Submod, com_Submod), in the assignable processes (acquisition, cognition, servoactivation, communication). The f_types correspond directly to the module classes presented in § 2.7 above.

§ 12.3 Function Classes within Module Code

This refers to functions within the software. Some code functions are specific (local) to certain types of modules, and they will run only on those modules. *These are the _mod* functions*.

Some functions can run on different types of modules, interchangeably. *These are the _mod[g] functions* where “g” denotes those module types on which the function can run.

Terminology	Denotes ability to be run on:
_mod	any EVA module
_moda	any “A” type (acquisition) module
_modc	any “C” type (cognition) module
_mods	any “S” type (servoactivation) module
_modm	any “M” type (communication) module

_modac, _modas, _modam, _modcs, _modcm, _modacs, _modacm, _modasm --- accordingly, as above

Some _mod functions can be shared across processors on other modules in a basic load-balancing type of parallel processing. In other words, the function will be executing in a multiprocessing manner, divided up according to how it is coded and in accordance with the operating system. This will be because of the computational tasks or for fault-tolerance capability building and assurance, but in principal they can be run on one single processor or processor-set on a single module. By definition, this capability overrides any restrictions in terms of module types; in other words, if it is multiprocessor-capable, then it can run on whatever processor is available on any type of module. *These are the _modmp functions* (“mod” + “mp” (multi-processor).

§ 12.4 Initial Taxonomy of EVA modules

Format:

- class_Name (class): string
- function_Type (f_type): string
- descriptor: text
- submodules
  - acq_Submod… (“acq_”, identifier) - may be multiple instances)
  - cog_Submod… (“cog_”, identifier) - may be multiple instances)
  - srv_Submod… (“srv_”, identifier) - may be multiple instances)

com_Submod… (“com_”, identifier) - may be multiple instances)

§ 12.4.1 Example: S-Water

S-Water
Flow Regulator 1 (flow_regulator_1)
Receive inputs from sensor(s) that measure hydration in one or more points (acq)
Control the flow of a liquid or viscous substance (e.g., water for irrigation) through a network of distribution devices such as pipes or tubes
Regulate the power supply
Control the power source
Regulate the water or other liquid supply
Collect data from sensor(s), power supply, liquid supply and structure data into standard representation format
Perform analysis of data on local module processor
Share local data with other modules in network
Receive data from other modules and incorporate that into

The representation in proto-code:

class: S-Water
f_type: flow_regulator_1
descriptor: Control of irrigation using one power supply, one water supply, one pump/regulator, one hydration (crop/field moisture content) sensor, one network of distribution pipes.
submodules:
- acq_hydration_sensor – determine if the irrigation network needs to deliver fresh water to crops
- acq_watersupply_sensor – determine level and sufficiency of the water supply
- acq_powersupply_monitor – measure the power supply (e.g., batteries) and if it is operating correctly
- acq_powergenerator_monitor – validate that the power generator device is operating correctly
- acq_pump_monitor – validate that the pump is operating correctly
- srv_power_control – regulate the flow of electric current into batteries and to the pump
- srv_pump_control – regulate the operation of the pump
- cog_irrigation_estimator – calculate the amount of water to be pumped into the irrigation network
- com_network_communicator – communicate data to and from the module with other modules

§ 12.5.1 EVA Module and Function Naming System

Grammar and Naming Conventions
We employ Extended Backus Naur Form (EBNF) as given by figure 3 below:

Usage	Notation
definition	=
concatenation	,
termination	;
alternation	|
optional	[...]
repetition	{ ... }
grouping	(...)
terminal string	" ... "
terminal string	' ... '
comment	(* ... *)
special sequence	? ... ?
exception	-

Figure 3 – Extended Backus-Naur Format (EBNF) Notation

§ 12.5.2 EVA Modules

The generic full description of an EVA module is:

module-name, superclass-identifier, class-identifier, [process-list]

[Example]

S-Water_M_001_(some process list)

There is a module known as “S-Water” which is for automated irrigation control. It is an M-type module. The first version of this type of module is class “001.” It has a set of processes that it performs, and this list can be derived from the list of submodules.

The full description is probably best implemented in XML. The short description can be more like the example presented above.

§ 12.5.3 EVA Functions

The generic description of an EVA function is:

[subsystem-specific-descriptor], descriptor, processing-type, argument-list

§ 13. More on CHANT and Banyan CSP

§ 13.1 Why "Banyan" as another name for CHANT, as a PDP/CSP system

The basis of the model is Communicating Sequential Processing (CSP). It is called "Banyan" because of the functional similarity to the tree of the same name, and how it propagates by the growth of descending tendrils that reach the soil and take root, forming entirely new "subnets" of the original arboreal growth. A CHANT system such as AgroIntel or any EVA system is designed so that it can grow and expand in much the same way as the banyan tree, without disturbance or complicated actions needed to be performed with the main system and its network, in order to add additional nodes.

§ 13.2 Subnets – part of the Banyan CSP Model

For example, one (1) single S-Water module constitutes a subnet. Within any subnet s can be multiple irrigation (distribution) units, and multiple sensors. Another S-Water or a different module can be added to subnet s, or it can constitute a different subnet, s'.

Here is how some the Adaptive Synthetic Intelligence ("ASI" as opposed to merely "AI") is going to work:

What are important in all the steps and functions are the Dynamics – the changes, the rates of change, the 1st, 2nd, even 3rd derivatives of functions (think: calculus!).

For each subnet
    For each module
        For each sensor-net
            For each sensor
                Interpret the data stream (validate) and normalize

All of this analysis is happening on the Arduino CPU (or some other processor) in each module and these are communicating their results, dynamically, in real-time.

The algorithm is ultimately "Parallel and Distributed."

Each sensor is normalized, with all the others in the set for that module, and then the same process is done for all the modules in the subnet. Naturally this can be a complicated set of calculations, because this "normalization" process must take into account many variances between different sensors or other data sources, for instance, which are not going to be very constant (e.g., real-time changes in locations). But once this process is done, in this manner we can automatically, autonomously, and with minimal user involvement, do the following (for example):

Adjust the control parameters that are used for regulating water flow and also optimizing the distribution of water among multiple channels (from the pumps). This can be particularly important if water, or electric power,

is at a premium. The same goes for fertilizers and other additives to irrigation or other distribution operations.

We can thus optimize and plan-ahead about needs for the crops, and also regarding the actual water supply (which may be limited, at times, especially if the climate is desert-like or semi-desert).

This document is only in its beginning stages. Now comes the time to define what is good, what is not, what is missing and incomplete, what is already done and should be incorporated into this architectural plan, what is done and needs to be changed.

The first objective is to get S-Water modules running in different agronomic settings and to have multiple modules working together as part of a singular, cohesive network. This and a lot more needs to be done.

Extended NOTES and References

[1]
In computer science, functional programming treats computation as the evaluation of mathematical functions with an avoidance of changing-state and mutable data. As a declarative programming paradigm, programming is done with expressions[1] or declarations[2] instead of statements such as in procedural (imperative) languages. In functional code, the output value of a function depends only on the arguments passed to the function. Thus, calling a function f twice with the same value for an argument x produces the same result f(x) each time; this is in contrast to procedures that depend on local or global states which may produce different results at different times when called with the same arguments but a *different program state*. In such procedural languages, the code within the procedures does not definitively indicate completely all the variants by which the procedure may have different outcomes, and this creates a situation that makes it much more difficult to ascertain all the ramifications of a given piece of code, and also difficult to manage the code for modifications and for prevention of errors. Eliminating side effects, i.e., changes in state that do not depend on the function inputs, can make it much easier to understand and predict the behavior of a program. This is one of the key motivations for functional programming.

Functional programming has its origins in lambda calculus, a formal system developed in the 1930s to investigate computability, the Entscheidungsproblem, function definition, function application, and recursion. Many functional programming languages can be viewed as elaborations on the lambda calculus. Another well-known declarative programming paradigm, logic programming, is based on relations.

In contrast, imperative, procedural programming changes state with commands in the source code, the simplest example being assignment. Assignment to a local variable within a block of code can depend upon many factors not solely internal to the procedural block (e.g., procedure, function, subroutine) or its arguments as passed by the invoking (caller) code. Imperative programming does have subroutine functions, but these are not functions in the mathematical sense. They can have side effects that may change the value of program state. Functions without return values therefore make sense. Because of this, they lack referential transparency, i.e., the same language expression can result in different values at different times depending on the state of the executing program. Tracking the source of the state-space changes can be difficult or virtually impossible.

Prominent programming languages that support functional programming include: Common Lisp, Scheme, Clojure, Wolfram Language (also known as Mathematica), Racket, Erlang, OCaml, Haskell, and F#. JavaScript has the properties of an untyped functional language, in addition to imperative and object-oriented paradigms. Functional programming is also supported in some domain-specific programming languages like R (statistics), J, K and Q, XQuery/XSLT (XML), and Opal. Widespread domain-specific declarative languages like SQL and Lex/Yacc use some elements of functional programming, especially in eschewing mutable values.
[above material taken from Wikipedia and other sources]

[4]
**** Total Modularity and Expandability is critical, essential, mandatory ****

YOU CAN BRING ANY FUTURE MODULE AND CONNECT IT AND THE REST OF THE NETWORK WILL KNOW ABOUT

IT AND KNOW WHAT TO DO.

- access and clear documentation for users

[8]
Some Abstraction-Level Remarks on the Overall Architecture

The following is extemporaneous, un-edited, freeform thinking about the overall Architecture.

We should closely examine how AgroIntel constitutes an Instance of an

Asymmetric Asynchronous Generalized MIMD (Multiple Instruction, Multiple Data) Parallel Processing Machine

What is this?

It is a network that can have plugged into it, or removed out of it, at any time, any one or a finite subset of the nodes that constitute the network and where the connections ("arcs") between any nodes can change and even be broken from time to time, and in all of this, the network (system) continues to function and is fault-tolerant and learning-capable.

But here I am first talking of abstract nodes, not only about physical processor elements and other devices. The network – the machine – consists of processes. And to execute those processes, there will be some configuration-set of physical devices (e.g., computers, sensors, servocontrollers, or complicated subsystems like robots).

(A process can also be thought of as an agent – thus, we are working here with multi-agent systems. We can use the term "cybot" to denote a given singular "agent.")

Any module that defines physical and computational processes, with some inputs and outputs and some state-space control model (the "program"), can be considered as a node within our network, and it can change in any of several ways:

- be activated (started)
- be terminated
- be modified, as it is running
- be modified by complete replacement (reload and restart)
- be impacted in its performance by changes in the type of inputs and type of outputs
- be impacted in its performance by the computational workload of itself and also other processors to which it is connected

For any change-situation in the network, a given node must know what to do. This includes:

- Here is some input and I do not recognize it, so I must give it to X which is responsible for this type of variance, for interfacing with this "new" (altered) type of input
- Here is some output and the recipient does not recognize it or respond to it, so I must give it to Y which is responsible for interfacing with such things
- Everything is processing too slow within my own node, so I must activate something with other nodes to take up the load and balance it accordingly, optimally, given what all the other nodes are doing presently.
- Everything is going too slow within some other node(s), so I must activate some alternative to compensate.

[11]

A Necessary Organic Model for Intelligent Systems

What is absolutely necessary, essential, required, today and tomorrow, for the survival and sustainability of civilization, is a thoroughly, totally ORGANIC approach to all of our Infrastructure Industries. This does not mean "organic" in the sense of "organic without fertilizers, etc." and certain agricultural and food production practices referred to historically as "organic." This means a biological model for information processing, knowledge representation and storage, and system integration, that is comparable in a formal and logical sense to how organisms manage their information and energy.

[12]

The Critical Importance of Control for Intelligent Energy Optimization and the Issues of "Over, Under and Random" in Human and Automated Control Systems

We should always be thinking about:

- Integration
- Complexity
- Optimization
- Re-use – the "Prius Flywheel" principle
- Smart control records and avoiding the problems of cyberattacks and ransomware as with EHR in USA and UK
- NomadEyes type sensor nets
- Stochastic random/mobile sensing, and use of SPSA and other randomized algorithms for energy optimization on small and medium and large (urban, national grid) scales basically the same way as with applications of the maths and models to Networks and to Surfaces (e.g., "wings with feathers")

