Zygmunt Ryznar
dr emeritus Cracow Poland
zygmunt.ryznar@pti.org.pl

OSL - Object Specification Language

(includes BSL,SSL,HSL,geometric structures & brain definition)

Abstract
OSL© is a markup descriptive language for a simple formal description of any object in terms of structure and behavior. In this paper we present a geometric approach, the kernel and subsets for IT system, business and human-being as a proof of universality. This language was invented as a tool for a free structured modeling and design.

Keywords
free structured modeling and design, OSL, object specification language, human descriptive language, factual markup, geometric view, object specification, GSO -geometrically structured objects.

I. INTRODUCTION

Generally, a free structured modeling is a collection of methods, techniques and tools for modeling the variable structure as an opposite to the widely used "well-structured" approach focused on top-down hierarchical decomposition. It is assumed that system boundaries are not finally defined, because the system is never to be completed and components are are ready to be modified at any time.

OSL is dedicated to present the various structures of objects, their relations and dynamics (events, actions and processes). This language may be counted among markup languages. The fundamentals of markup languages were created in SGML (Standard Generalized Markup Language) [6] descended from GML (name comes from first letters of surnames of authors: *G*oldfarb, *M*osher, *L*orie, later renamed as Generalized Markup Language) developed at IBM [15].

Markups are usually focused on:
- documents (tags: title, abstract, keywords, menu, footnote etc.) e.g. latex [14],
- objects (e.g. OSL),
- internet pages (tags used in html, xml – based on SGML [8] rules),
- data (e.g. structured data markup: microdata, microformat, RFDa [13],YAML [12]).

A markup is an annotation (tag, label) that is syntactically distinguishable in the contents. There are several types of markup:

1.descriptive markup used to label parts of the document written in LaTeX, HTML (Hypertext Markup Language) and XML (Extensible Markup Language),
2.presentational markup in form of hidden binary codes embedded within document text and dedicated to produce the WYSIWYG effect.
3.procedural markups e.g. names of macros or subroutines to be invoked by program,
4. factual markups that are integral part of contents e.g. phrases of OSL

Specification languages differ in many ways (notation, scope, presentation etc.). OSL is nongraphical conceptual language so it is very different from such diagrammatic language as Unified Modeling Language UML[10] even if UML is enriched with textual Object Constraint Language (OCL) [11]. We have no objections against graphical visualization - it could be assumed that diagramms and geometric figures could be generated from OSL expressions at the implementation stage using tools like IDL [6].

OSL has been built as open and flexible. It covers not only main objects (subjects) but also whole environment all around defined globally (at kernel) and locally (at subject level). There are no borders of definition. One could imagine description of universe or more (let's call it "galaxy") even. OSL includes also open objects which could be incorporated in any "place" like free electrons from atom.

Another feature of OSL is an object orientation. Objects may be concrete (physically existent) and abstract (conceptual, virtual). The physical objects are viewed ontologically. Any object has at least its own name, identifier, properties, structure, interface and relations to other objects, own behaviour and "life-history".

OSL does not concentrate on data structure and documents but on descriptive statements in a classical wording specification (with geometric conceptual figures) focused on the layout and behaviour of objects.

A geometric view has two profiles. Profile 1 expresses standard OSL geometric view and consists of many conceptual geometrical figures. Profile2 includes flat figures like flowcharts showing input-output and function or program (in computer job sequence) and diagrams presenting the structure (e.g. class diagram in UML[9]. These profile2 items are of classic type and can be easily made manually or generated from detail specification.

A geometric view could be an inspiration for new type of data in the cognitive computing or bigdata mining, as geometric shapes may be discovered by analysis of various massive information. We assumed that geometric interpretation might be be more accurate and more evocative because it shows imaginably (particularly in case of spiral) the nature of objects at least in terms of space and time. Here, it should be mentioned, that fuzzy sets technology uses a geometric view (trapeze and triangle functions).

There are many geometric figures. Some of them are uncommon (swarm, bunch, blackhole, wormhole, freespace) and become a challenge to be taken up in conceptual modeling.

1. <u>Spiral</u> differs from classic iteration in that sense, that every scroll can possess different “engine” and content. Quite “powerful” are multistrand spirals. Examples of strands in business could be profit, competitiveness, credit ability etc. Such spiral in a business may illustrate for example the bankruptcy of company which falls into spiral that pilots call the “death spiral”. A special type of spiral is an irregular hyperbolic spiral intersecting an asymptote infinitely many times.
   Spiral approach is known in software development where the spiral has four phases (planning, risk analysis, engineering, evaluation) and project repeatedly passes through these phases in every iteration.

2. Swarm relates to the homogeneous relocatable dense population. The path of moving swarm is a good illustration of global expansion of business or moving a business from one country to another.
3. Free space means unstructured or with no rules of structuring.
4. Tunnel is three-dimensional population with the value X (e.g. the sum of credits) along the time Y and densed acording to the third factor Z (e.g. credit ability).
5. Cylinder differs from the tunnel with the feature that it carries the values on the surface while the tunnel keeps them in the interior.
6. Triangle coud represent "shadows" of object on its 3 angles surface (a-axis, b-axis, c-axis), e.g. for employees population angles are age, education, sex. The father of a triadic approach was famous philosopher G.W.F.Hegel (1770-1831) who used triangle to visualize a 'system of science' as a triangle with angles: logic, nature, spirit.[19]
7. Blackhole marks the irreversible disappearance of the object (e.g. the bankruptcy of the company) and shows the „strength of drawing in" (e.g. a speed of the bankruptcy).
8. Darkbox is a place for a dark (or hidden) information.
9. Cloud means an external container equipped with secure gate to enter it.
10. Neural networks are networks of interconnected layers and nodes, which process information as a response to external inputs using methods unavailable in traditional calculations. Very known applications of neural networks are character recognition, human face recognition and signature verification. In business they are used when solution is not based on a predetermined and preweighted criteria but on the past experience (e.g. in banking area - failures in loan granting, forecast time dependent variables such as net income for each month of a year. Neural networks are applied also in finding trends in large quantities of data (currency, stock exchange prediction).
11. Fractals
    a) fractal geometry in multifractal stochastic volatility models that work in the context of dynamic turbulance used for example for modeling market fluctuations. [7].Fractal analysis can roughly be thought of as a way to characterize and investigate the properties (e.g. selfsimilarity) of irregular sets.
    b) fractal networks useful to express fractal populations like franchising networks.

II. OSL NOTATION [1]

<!...>	comment
< >	container of phrase
≡>	link to something external (outside area)
<def > </def>	start-end of definition
<spec> </spec>	start-end of specification
<beg> <end>	start-end of body section
[..] or {..} [2]	list of assigned keywords
(..)	list of items
xxxx(..)	name of list
=	value assignment
@	mark of special attribute,feature,property
@dark	unknown, to obtain, to discover

1 Notation and kernel of this version differ considerably from ones published in [3].

2 If using Latex editor we suggest [..] brackets instead of {..}.

::	belongs to
:	equivalent name (e.g.shortname)
#	number of
|name|	executive/operational object
ppppXxxx	name of item Xxxx with prefix ‘pppp ‘
XXXX	basic object
UUUU.xxxx	xxxx object belonged to UUUU object class
& /	conjunctions ‘and’ ‘or’
<->[3] or ↔	Bidirectional unitary relation 1:1
<- or ←	Back directional unitary relation 1:1
-> or →	Forward directional unitary relation 1:1
<=> <= =>	Many to many relations
..	More
...	Much more

III. OSL KERNEL

The kernel contains standard phrases and keywords common for all areas (subsets) and environment. A special attention in the kernel is paid to relations between objects. Jay W.Forrester author of fundamental work "Industrial Dynamics”[4] appreciates the importance of relations in [18]: “the structure of interconnections and the interactions are often far more important than the parts of system”.

```
<def OSL>
    <def ENVIRONMENT: ENV>
         ENV[regulations, infrastructure:INFR]
         regulations[legalacts,resolutions,decisions]
         INFR[it,org,hr]:[itINFR,orgINFR,hrINFR]
         itINFR[servers,opersystems,applications,databases,users,prlanguages]
         orgINFR<!org. structure of subject>
         hrINFR<!human resources>
    </def>
    <def globalMapping>
        def subLang[BSL,HSL,SSL]<!subsets of OSL>
        objList<!list of objects>[area,subject,problem,decision,<defined objects>]
        defList<!list of definitions>
        specList<!list of specifications>
    </def>
  class<!class tree>

 <!object definition>
 <def subject <NAME><!main object name>
 <def <name><!ordinary object/item name>
  object.id<!object identifier>
  object.type[eObject<!elementary atomic object >,
              dObject<!dynamic object >,
```

[3] If typing on keyboard

```
               iObject<!informational object >,
               vObject<!virtual object >,
               sObject<!smart object)>,
               oObject<!open object>,
               incObject<!incarnation of object>,
               binObject:BINDER<!collection of integrated objects>,
               copyObject<!copy of object>
               probObject:PROBLEM<!task to be performed>,
               interObject<object created by intersection of objects>,
               capsObject:CAPSULE<!portion of information reserved for a given user>]
                interObject[(list of objects) when <condition>]
               sObject[noiceReduction,selfTeach,selfRepair,selfKill,selfRestore,selfRestart]
               oObject[input(parameters,data),output(info,messages),
                        structure(addComponent,addRelations)]
</def>


<!dynamics definition>
  event:ev<!-elementary atomic fact >
  operation:op
  action:ac<!sequence of operations or events>
  process:pr<!sequence of actions and events>
  pr[trigger,<actions><events>,endEvent]
  dynamics[event,operation/transaction,action,process]
  dynamics[ev,op/tr,ac,pr]<!short notation>
  dynamics.scenario[evSc,opSc,acSc,prSc]<!event scenario,….>
  trans<!transaction in terms of operating system monitor>
  ftrans<!financial transaction>
  reverseMode[rev,rAc,rOp,rTr]<!back to the previous state>
  scenario:sc<!predicted sequence of actions and events>
  scenario.rank[best,middle,worst]
  object.Info<!information visible at the moment of access>
  keywords:kwords<!additional keywords in def>
  olh<!object life history>[timeline,events,aging-curve]

<!interactions-relations>

  role[interface,integrator,component,monitor,commander,
       driver,trigger,reactor,agent,executor,generator
       locator,executor/performer,initiator,terminator,destructor,
       participator,owner,stockholder,customer,supplier;partner,employee]

 relations[activated by,activates,assisted by,built from ,
           appearence depends on ,belongs to/is owned by ,
           exists as satellite of <object>,calls <object> (<interface>),
           consists of <parts>,contained in/contains,
           controlled by/controls,derived from,
           existence depends on,exists when/in/for,
           included in,linked to ...by/links,
           refers to,relates to,related by affinity,
           represented by/represents,involved in,
            shared by/shares,used by/uses]

 state[active,inactive,dark,dormant,suspended,aborted,
       variable,invariable,idle/waiting,lost,expected,deleted,homeless]
 status[generic,real,virtual,undefined]
 reactor[acceptance,rejection,constructor]
 rank[critical,necessary,most wanted,optional,worst,best]
 rule[decision-table,when-if,formula].

 layout[shape(gProfile1),gProfile2,sparcity,density,variability]

<def gProfile1><!standard-geometric profile>
      [free-space,swarm,bunch,network,neural-network,hierachy,line,triangle,tunnel,
       curve,spiral,spring,circle,elipse,cylinder,sphere,ellipsoid,con,doublecon,prism,
       fractal,fractal networks,squarepiramid,container,blackhole,wormhole,cloud,darkbox]


   <def spiral>
     spiral[single-strand,multi-strand]
     spiral[helix,logarithmic,hyperbolic,polygonal,rational,golden,
          spherical,conical,circle-involute,cornu,daisy,epispiral,
          archimedian,fermat,nielsen,ulam,poinsot,phyllotaxis]
      helix<!a curve for which the tangent makes
          a constant angle with a fixed line>
      spiral.parameters[center-point,number-of-rotation,
          number-of-band,starting-radius,points-per-rotation,
          growth-per-rotation(horizontal,vertical)]
    </def spiral>

     spring<!simple iteration>
     swarm<!moveable homogeneous population with variable density>
     bunch<!nonmoveable homogeneous population>
     circle[edgeCircle<!population on the edge>,
            insCircle<!population on the surface>]
     elipse[surElipse, edgeElipse]
     cylinder[edgeCylinder,insCylinder:tunnel]
           edgeCylinder<!population on the edge of Cylinder>
           insCylinder<!population inside of cylinder>
     triangle<!ayout defined by 3 factors always existed and related together>
     container<!trunk, regular 3-dimensional figure)
     blackhole<!"off the face of the surface">
     wormhole<!place injected with foreign/strange information>
     free space<!no limits, no predefined structure)
     line[single,multiline]
     curve[parabola,hyperbola,….]
     point<!something that may exist only as a single event e.g. big bang>
        objPoint<!single event for a given object e.g. birthday>
    solids[sphere,cone,pyramid,cube,cylinder]
    polygons[rectangle,square,pentagon,hexagon,octagon]
   <def  neural-network>
       neural-network.type[singlelayer,multilayer,Kohonen,Hopfield,convolutional]
       neural-network.parameters[layers(input,intermediate,output)(hidden/visible),
       connection-between-layers(backpropagation,...),variables,expected-values,
       weights,rules-for-modifying-weights,learning-method][4]
   </def  neural-network>
</def  gProfile1>

<def gProfile2><!flowchart,diagram>
  [flowchart(prFlowchart<!program functions>,jobFlowchart<!sequence of programs>),
   diagram(<flat structure diagrams>)]
</def  gProfile2>


<def control-flow>
    ac(ev1,ev2,ev3, ..)<!action-sequence of events>
    pr(ac1,ac2,ac3,...)<!process>
    s(ev1,ev2,ev3, ..)<!sequential flow of events>
    p(ev1,ev2,ev3, ..)<!parallel flow of events>
    pr(s(ac1,s(ev1,ev2,ev3),ac2(p(ev4,ev5,ev6),(ev7,ev8,..))<!mixed flow>
    repetition[algebraic-iteration,spring,spiral]
    activated by <..> with <initial-value> at <time-point>
                  when  <condition>
    finished at < > with <...> when  <..>
 </def>

  <def  body>
      body[Contents,Script,gprofile]
      contents<!e.g. document, program code>
      script<!script generated   upon the pattern of   behaviour >
      <beg><!sections of  body>
           <beg>
             …..
           <end>
      <end>
   </def  body>
</def  kernel>
```

An OSL extension for selected areas is defined in subsets: OSL-B(business), OSL-H (human-being), and OSL-S (systems). A subset is similar is a shell assigned for a group of users allowing them to use its own commands and keywords.

4We show here   basic features of neural networks to show OSL capabilities. An extended specification of it is a task for experts.

IV. OSL-S FOR SYSTEM

Subset OSL-S is dedicated to IT system specification and is oriented on the free structured design aimed to making adaptative software for ill and well structured problems. The essense of this approach is a creation of the library of building blocks, tuning them and assembling into application packages according to actual needs of users. So this method prefers bottom-up aproach but the integrity is provided by good specification of objects and problems (made mainly by top-down approach).

One way to build flexibility is designing skeletal programs that have standardized (but multifunctional) control flow and are equipped with many "modifiers" that should be tuned before use. The tuner may perform such operations as inserting data names and parameter values, choosing entry points, generating CALL statements, generating empty modules (driver or stub type), inserting expressions into macrostatements and invoking database schema.

There are following steps in a free structured design:
1.specification of objects (conceptual level)
2.preconstruction – creating of building blocks library (physical level)
3.problem driven design – problem decomposition of up to the tuning requirements and assembling them at logical and physical levels
4.making the software package which meets the problem requirements (physical level)
5.iteration (1-4 steps) – and adaptation according to changes in the problem.

```
<def SSL>
<def subject  itSYSTEM>

  itINFR[servers,operSystems,applications,transDataBases,dataWarehouses,
         clouds,networkMgtSystem,users,prLanguages]
  <NAME>[SYSTEM.subsystem.module.program.pr-block]
  pr-block<!building block,generic program/subprogram>
  kwords[version,interface,run,runTime,integrated,standalone,inDevelopment,accepted,
         notAccepted,library,creatDate,updDate,tested,rejected,pcode,ecode,callValue,
         trace,errorCode,inItems,outItems,flow,read-only,cloud]
  trace[path,callValue,outValue,errorCode]
  process[trigger,action(<events>),endEvent]<!when process is invoked simultaneously by many
  programs each instance is recognized by pcode,  each event by ecode>

<def  pr-name>
     object.Info[pr-name,version,author,creatDate,updDate,prLanguage,operSystem]
   <def control-flow>
         refers to <flowchart-name>
        activated  by  <program/procedure-name> with
       <initial-value> at <time-point > when   <condition>
        finished at (< time-point /no-of-repetition>,<date>)
        runs for  (<time>,<time-intervals>,for-query)
        with  <value/output> when <condition>
        condition [forAll,forFirst,forLast,for#]
   </def>
</def>
```

```
<def  PR-BLOCK(name)<!reenterable ProcName)>
      objectInfo[name,version,author,updDate,codeSize,prLanguage,operSystem]
      resources[dataBuffer,eventTrace,stackHandler]
    <def  flow>
            call
           ac(verif)  <!verification]
               ev(checkPassword,callVerif)
                    when callVerif  failed exit
           ac(initial) when first call
               ev(bufferDecl,stackDecl),
          act(tuning)
              ev(paramAnalysis,transform,generateExecutable)
          ac(activate)
               ev(paramAnalysis,tuner),
          ac(run)
                ev(load,perform,releaseResources,exit)
           </def>
        <def interface>
          <!at run-time interface retains an actual  state of resources  for each call>
             call[<callingName>,<calledName>,<tunerName>
                   <password><!optional>,<entryPoint>,
                  (<parameters,modifiers>),inItems,outItems]
         </def>
  </def>
```

V   OSL-B   OSL FOR BUSINESS

Subset OSL-B named **BSL** (Business Specification Language) is   focused on business objects. There are many businesses. The widest spectrum of activities exists in   manufacturing industry. In [4]   are specified six main flows: material, energy, ,orders,finance, human resources and information. Information flow links all streams giving   the overall picture of enterprise.
Banking belongs to very complicated business if taking into account not only simple products like accounts, deposits and loans but also derivatives, forex, flow cash projection and risk management.
In our specification we present some sections of this activity   for illustration   only. The full specification   should contain also such objects   as headoffice, branch, channel of product delivery, many products and types of transactions, executive operational objects, like account manager, teller, dealer and IT infrastructure.

<**def BSL**>

BUSINESS[BANKING,INDUSTRY,TRADE, SERVICES]

 <**def subject** BANKING>

```
BANKING[RETAIL,WHOSALE,UNIVERSAL,MONEY-MARKET,DERIVATES,SHARES]
<def ENV>
     bank.id(BIC<!Bank Identification Code>,
     account.id[IBAN<!International Bank Account Number>,delivery-channels)
     itSystem(<system.subsystems.modules>)
     dataTables[Libor,OperatingCurrences,ExchangeRates,
    delivery-channels[internet-accounts(computer,smartphone,iTV),phoneline,branch,ATM]]
</def>

kwords[customer.id,account,listAccount,accountCurrency,creationDate,cashFlow]
      fTrans<!financial-transaction>,fTransLimit]

problem[capital-assets-level,new-product-demand,customer-satisfaction]

<def BANKING.retail>
        retail.product[currACCOUNT,DEPOSIT,LOAN]

 <def subject BANK>
        object.Info[BIC,country,bCurrency<!base currency>,
                    FinancialYear,#branches <!number of branches>]
        dataTables[corrBanks,Branches,calendarWorkingDays,
                   bkAccountChart,productList,interestRateTable]
        kwords[branchNo,idCustomer,accountNo,rate,balance,balanceSheet]
        OperationalObjects|teller,accountMgr,customerMgr,productMgr,trader|
        Bank.objects[product,currency,limit,account]
        limit[country,industry,customer,currency]
        cluster[profit-rate,growth-rate,competitiveness]<!Global factors for subject>
        iObject[customerPosition,monthlyBalancesheet]<!for each customer>
        typeBank[dmBank<!domestic>,frBank≡><!foreign>,corrBank≡><!correspondent>]
        bkAccount[bsAccount<!balance-sheet>,nbsAccount<!nonbalance-sheetAccount>]
        batchOperations[eodOperation<!at end of day>,eomOperation<!at end of month>,
                   eoyOperation<!at end of year>,eopOperation<!at end of product>]

 <def currACCOUNT><!current account>
         objectInfo[Account id,owner,co-owner,minBalance,actualBalance,historyStatement]
         Relates to idCustomer
         rtTrans[Open,Quit,Cash-in,Cash-out,transfer]<!real time transaction>
         eomOperation[monthlyStatement ]
   </def>

   <def CUSTOMER>
         Belongs to <customer-segment> evaluated by dataMining/neural-network
         listAccount:(<list of accounts>),
         LOAN activated when accepted,
         customerPosition(#account,cashFlow,overDrafts)
</def>
 </def  subject>
 </def Banking.RETAIL>
```

<u>other areas in a business:</u>

```
<def AREA(INDUSTRY)><!to be defined>
       <!objects, material flow,energy flow,order flow,
         financial flow,information flow,HR>
</def>
```

```
<def AREA(TRADE)><!to be defined>
      <!objects, services flow,
        financial flow,information flow,HR>
</def>
<def AREA(SERVICES)><!to be defined>
     <!objects, services flow,
        financial flow,information flow,HR>
</def></def  bsl>

  <spec Banking.RETAIL(IndustryBank)<!simplified example>
        BIC=ALBPXLPW
        customer.id=XXXXXXX
        current.account.id=PL 99 9999 9999 9999 9999
        owner=John Stale
        co-owner=Jane Stale
        baseAccount.currency =USD
        creation.date=.<...>
        transaction-limit=<...>
        info.set(account) <balance and list of transactions  for each account>
        info.set(products)<list of accounts>
        info.set(channels)<branches used,echannels used>
   </spec>
```

VI. OSL-H OSL FOR HUMAN

Subset OSL-H named **HSL** (Human Specification Language) is a semiformal notation for a human being and may be a tool dedicated to professionals dealing with human resources or anyone interested in psychotechnology. Further development of HSL toward psychology and medicine could be achieved with close collaboration with psychologists and medical professionals.

Human being is the most important object of system and himself may be treated as an "open system which maintains a constant state while the matter and energy which enter it keep changing"[5 p.11].Human plays many roles in decision taking, execution, communication etc.

One possible usage of HSL language is creation of a human resources database in a corporation or even on an international scale for locating individuals which meet certain psychological, intellectual and professional requirements. Particularly it could be useful to form well collaborating teams in process of system analysis, design, programming and implementation.

Similarly to other subsets of OSL this one contains only additional phrases and keywords that do not exist in the OSL kernel. A scope of human specification may be expanded by many other interesting topics like „human thought - the physiological process of mentation" and „ bodily and facial gestures as a factor in communication" included in HUML (Human Markup Language) [17].

```
 <def HSL>
 <def subject HUMAN>
        class1[animals.mammalia.primates.homidae]
        class2[nation.ethnic-group.profession.person]
        kwords[life-space,behaviour,scope]
        scope[biophysical,geogr,cultural,social, legal]
 <def ENV><!environment>
       ENV[WORLD,CONTINENT,COUNTRY,REGION,SITE]
       ENV.legal<!Legal acts, resolutions, decisions>
       ENV.cultural[tradition, history, education,religion, ideology, art, radio-tv]
       ENV.biophysical[animals.homosapiens]
       ENV.geogr[homeAddress,company/school]
</def>
<def BRAIN>
   brainId(personId &/biologicalId,type)
   mainparts[forebrain(cerebrum(hemisphere,thalamus,hypothalamus),midbrain,hindbrain]
   area[lobe,cortical-region]
     lobe[frontal,parietal,limbic,occipital,temporal]
     cortical-region[primary-visual,entorhinal,inferior-temporal,orbitofrontal,
                     lateral-prefrontal,inferior-parietal]
     cortex[visual,sensory,auditory,..]
   parts[cerebral-cortex,basal,dienceph,brainstem,cerebellum,hippocampus,spinal-cord]
   brain-function[sensory(vision,hearing,smell,touch,...),
                  mental(association,speech,language-comprehension,coordination,...),
                  motor(eye-movement,voluntary-movement, ..)]
   detail-parts[neuron,synapse,receptor,unpaired-electron,neurotransmitter,..]
     neuron{[nucleous(mithochondria,membrane,cytoplasm,vesicle,perycarion),
             myelin-sheath,schwann-cell,axon,dendrite],
             form(multipolar,bipolar,unipolar)]}

   view[active/passive (biological,chemical,energic,geometric,physical,medical,
           logical,semantic,psychological,mathematical,ontological)]
           <!active view is based on brain neural-network>
   engine[thinking,emotions,info-retrieval,memorizing,intuition(trust,love,hate),
             communication,total,......]

   emotionEngine =>  psychologicalView
      emotionEngine[Lob.frontal(electrode-placements/neural network-area)]
      psychologicalView[love,hate,satisfaction,frustration,agression,enjoyment,anger,
                        fear of insupport,regression,inferiority,persecution] </def>
    totalEngine =>  medView
      totalEngine(brainArea/neural network-area,disease-pattern)
      medView(disease(neurodegenerative,neurological,..),injury)
            neurodegenerative(dementia/alzheimers,parkinsons,huntingtons,..)
            neurological(autism-spectrum,tumor,migraine,multiplesclerosis,epilepsy,stroke,..)
     totalEngine =>/ → ontologicalView
      totalEngine(brainStructure,brainContents)
      ontologicalView(brainId,brainAging(neurons(dead,born)),diseasesHistory,
                               brainVolume(curve,...),brainUsage)
     totalEngine =>/ → energeticView
       totalEngine(brainStructure,brainContents)
       energeticView(brainId,power-consumption vs aging,power-supply-disturbances)

</def brain>

 <def  PERSON>
       object.nfo[id,sex,birth-data]
       invariables[id,sex,birth-data]
       homeaddress[country,site,street,house,flat]
       sex=(male/female/x ]
       body[(brain,liver,kidney,joints,..) weight,height,eyes-colour,defects]
       family[gentree,parent,child,son,daughter,
             grandSon,grandDaughter,granMa,granPa]
       emotion[love,hate,satisfaction,frustration,agression,enjoyment,anger]
       psychComplex[fear-of-insupport,regression,inferiority,persecution]
       habit,hobby,profession,
       health[measures,physical-examinations,illness-history],
       role[advisor,consultant,manager,patron,partner,customer,
           supervisor,participator,owner,supplier,
           user,analyst,designer,programer,operator] <!plus 'role' in kernel>      ,
       appearence depends on,assisted by,belongs to,matched/matches,
       relations<!plus relations  in kernel>
       relates to <family-members>used by,uses,not used,misused,abused,
       state[active,inactive,dormant,suspended,aborted,idle,lost,dead,
            homeless,retired,married/divorced/single,ignored]
       place[point, area,everywhere,nowhere]
       life-space[psychological,social,educational,professional,financial]
       behaviour<!flow of processes of the object >
       behaviour.rational[selfrealization,need,satisfaction]
       behaviour[marriage,friendship,career,ilness,aging]
       genotype,fenotype
       olh:=[birth,aging-curve,social_events,health_illness-events,
            educ-events,job-events,critical_events,death]<!object-life-history
<def> cluster<!GlobalFactor estimated on the base of several particular factors>
       cluster[self,profile/type,attitude,leadership,ability,
             extraversion,anxiety,independence,healthState,
             lifeStyle,creativePotential,happiness,BipolarPersonality]
       self[self-identity,self-assesment,self-sentiment,self-esteem,
            self-regard,self-reliance,self-control,
            self-image,self-extension,self-structure]
       leadership[assertive,creative,facilitative,independent,
                    stable,permissive,leadership(Style,Potential]
       ability[toughMinded/openMinded,creative,fast/slow,
             toleratesDisorder/perfectionistic,grounded/abstracted,
            improving own learning,problem solving, IQ, ......]
       need[biological(food,medical,emergency,rescue, coping),
           cultural,psychological(love,esteem,selfrealization),
           financial-resources,security]
```

```
        BipolarPersonality[Warmth,Reasoning,EmotionalStability
                           Concillation,Dominance,Liveliness,Openness,
                           Tension,Rule-Consciousness,SocialBoldness,
                           Sensitivity,Vigilance,Abstractedness,
                           Privateness,Apprehension,OpennessToChange,
                           Self-Reliance,Perfectionism]
</def>
<def> BipolarPersonality
          Warmth(reserved/warm)
          Reasoning(concrete/abstract)
          EmotionalStability(emotional/stable)
          Concillation(concillatory/aggressive)
          Dominance(deferential/dominant)
          Liveliness(serious/lively)
          Openness(extraversive/introversive)
          Tension(relaxed/tense)
          Rule-Consciousness(expedient/rule-Conscious)
          SocialBoldness(shy/socially-bold)
          Sensitivity(utilitarian/sensitive)
          Vigilance(trusting/vigilant)
          Abstractedness(grounded/abstracted)
          Privateness(forthright/private)
          Apprehension(self-assured/apprehensive)
          OpennessToChange(traditional/open-to-change)
          Self-Reliance(group-oriented/self-reliant)
          Perfectionism(tolerates disorder/perfectionistic)
  </def>
</def >
<def  IT.TEAM> <!this definition may be a part of OSL-S>
     member[user,analyst,designer,programer,tester,consultant]
      member.requirements[perceptive-listener,communication-
          skills,strong-interest-in-job,persistent,having-stamina,
          disciplined,creative,open-minded]
</def>
 </def  HSL>
```

VII. CONCLUSION

The scope of this paper covers conceptual and physical modeling of the system including the interfaces between a system and its environment and psychological factors concerning its interfaces with humans.

OSL is a factual markups language for defining the „object-world" in terms of the structure, behaviour and relations. An interesting feature of it is modeling using GSO (Geometricallly Structured Objects).This language can be used for specification of business, IT systems and any other area.

OSL-S (OSL for System) may document core features of IT system before, during or/and after design and - if implemented - would be a tool of ASF (Automatic System Factory). It shows a way how to create systems with variable structure using a library of tuneable blocks. So in this case OSL could be a "hybrid" solution both for the specification as well as for execution.

OSL-B (OSL for Business) may cover variety of businesses in terms of information flow, procedures, products and financial flow.

OSL-H (Human Specification Language) would be useful for HR services in terms of precise search of specialists with certain psychological and professional characteristics.

Implementation of OSL requires to build dictionary of keywords and phrases, database of objects, tools for converting geometrics and relations to graphics to present overall picture of relations between objects. We assume that presented here a conceptual (non-grahical) geometric view may be more accurate and flexible in many cases.

References

articles:

[1] author = Ryznar, Z.
title = A conceptual model of an interfunctional data base system,
journal = Information and Management,
year = 1978, volume = 2, pages = 67–74,

[2] author = Ryznar, Z.
title = S&DL – Structured Design Language,
journal = Angewandte Informatik-Applied Informatics,
year = 1981, volume = 12, pages = 526--533,

[3] author = Ryznar, Z.
title = OSL Object Specification Language,
journal = Journal of American Academic Research JAAR,
year = 2017, volume = 5, pages = 47–52,

books:

[4] author = Forrester J.W.
title = Industrial Dynamics,
address = USA,publisher = The MIT Press, year = 1961,

[5] author = Johnson R.A.,Cast F.E.,Rosenzweig J.E.
title = The Theory and Management of Systems.,
address = USA,publisher = McGraw-Hill Book Co, year = 1967,

[6] author = Bowman K.P.
title = An Introduction to Programming with IDL: Interactive Data Language ,
address = USA, publisher = Elsevier Academic Press, year = 2006,

[7] author = Mandelbrot, B.B.,
title = Fractals and Scaling in Finance:Discontinuity, Concentration, Risk,
address =New York, publisher = Springer-Verlag,: year = 1997

internet links:

[8] www.w3.org/MarkUp/SGML/

[9] www.cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Applications/daveapps.html

[10 www.omg.org/spec/UML/2.5.1

[11] www.*omg.org/spec/OCL/*OCL Specification v2.4.1 (2014).

[12] www.yaml.org/spec/

[13] webdesign.tutsplus.com/articles/an-introduction-to-structured-data-markup--webdesign-8577

[14] www.latexeditor.org
[15] www.sgmlsource.com/history/roots.htm
[16] www.harrisgeospatial.com/SoftwareTechnology/IDL.aspx#graphics
[17] www.oasis-open.org/committees/human markup/documents/HM.Primary-Base-Spec-1.0.html
[18] pubsonline.informs.org/doi/pdf/10.1287/mnsc.14.9.601
[19] www.hegel.net/en/e0.htm

Post Scriptum.

This paper is published to encourage any person or research institute to continue the OSL project towards the implementation. This work has not been supported by any organization and may be used under Creative Commons – 3.0 Licence.