Zygmunt Ryznar

HSL - Human Specification Language (proposal)

last update 1 April 2015

Who are we? Who do we think we are?

Presented here proposal for a human definition is dedicated to anyone interested in ontology and existential psychology.
HSL - Human Specification Language (you may call it "Human Definition Language" - HDL) is a semiformal notation for a human being.
The topic "human being" is not a simple one. Several psychologists and writers (G.Lindzey, 1974; D. Schultz, 1986; R. Corsini, 1977, Atwood and Tomkins, 1976) have stated that the several major theories on personality “are inevitably colored by subjective factors determined not only by certain empirical facts which are generally agreed upon, but also by a whole range of idiosyncratic factors and motivations affecting each theorist as an individual”.

My main profession is an information system analysis and design. During many years of communication with business and IT people I have got the conclussion, that there is "a must" to have a tool for precise specification. So, I created OSL (Object Specification Language), as an universal tool for formal description of any ontological (e.g. human being) or quasi-ontological object (e.g. banking account). HSL is a subset or a variant of it.
One can take HSL specification for further development upon condition that source will be shown and a message sent to zygmunt.ryznar@gmail.com

Język HSL jest półformalną notacją opartą na predefiniowanym słownictwie, której celem jest sformalizowanie opisu człowieka w wielu aspektach. Docelowo uściślenie tej propozycji można osiągnąć przy współudziale odpowiednich specjalistów z dziedziny psychologii i medycyny. Po implemetacji języka (m.i. poprzez opracowanie konwertera opisu na struktury danych) i załadowaniu bazy danych (wyniki badań, wywiady, ankiety...) możliwe wydaje się obliczanie tzw. czynników globalnych, charakteryzujących daną jednostkę (np. pod względem osobowości, potencjału twórczego, stanu zdrowia itp.).
HSL obecnie posiada zdefiniowane podstawowe zwroty. Potem zapewne przyjdzie pora na techniczną "obudowę" m.i. w postaci schematu danych (np. jako XMLSchema - dtd lub xsd), w oparciu o który utworzyć można bazę danych. Obecna wersja języka nie obejmuje na razie powszechnie znanych danych umieszczanych w tzw.kwestionariuszach osobowych, gdyż odbiegają one od głównego profilu HSL i tylko kwestią techniczną jest późniejsze ich dodanie.

Notacja pod względem formalnym jest stosunkowo prosta i nie nosi charakteru matematycznego. Ze względu na róznorodność zwrotów jego opanowanie zapewne wymagać będzie pewnego wysiłku. Poza "standardami" starałem się wprowadzać do notacji takie oryginalne rozwiązania jak np. geometryczna prezentacja
- repetition :=(iteration, single spiral, multiband spiral)
- population layout := (free-space,swarm,network,hierachy,line,triangle,tunnel...)

Dzięki anglojęzycznej składni HSL baza danych nosić będzie charakter międzynarodowy. Budowana może być jako baza globalna lub lokalna w ramach organizacji, firm lub krajów, a jednym z jej celów może być wyszukiwanie osób spełniających określone kryteria.
Definicja człowieka nie jest prostą sprawą, o czym świadczyć mogą np. liczne teorie osobowości - "Functional autonomy" Allporta, "Basic concepts for a psychology of personality" Murray'a H.A ,"Trait theory", "16 Personality Factors", "9 Enneagram types", "Myers-Briggs Type Indicator", etc. Wydaje się, że można je - po pewnym uściśleniu lub ujednoliceniu terminologii - "włożyć" do opisu w HSL, zapewniając dzięki temu szerszą bazę porównawczą, a w następnym kroku próbować ją ujednolicić wg wspólnych kryteriów. Ale to jest zadanie dla psychologów a nie dla informatyka, który daje tylko narzędzie.
Several psychologists and writers have (G. Lindzey, 1974; D. Schultz, 1986; R. Corsini, 1977, Atwood and Tomkins, 1976) stated that the several major theories on personality “are inevitably colored by subjective factors determined not only by certain empirical facts which are generally agreed upon, but also by a whole range of idiosyncratic factors and motivations affecting each

theorist as an individual"..

"There's nothing as practical as good theory" *(Kurt Lewin)*

HSL - HumanBeing Specification Language - basic notation - *ver. 0.4*

HSL is a subset of OSL (Object Specification Language) dedicated to define ontological objects. Within HSL an ontological object is independent (self-contained) object having at least its own name, identifier, properties (traits), behaviour and "history". Object class includes various types of objects (concrete and abstract, existent and non-existent, real and ideal, independent and dependent).
OSL specification language was primarily developed for business objects-example and afterwards adapted for human-example.
Because environment and properties are different in each case, there was a need to add some keywords and subclasses for Human specification and they were derived from existential philosophy, existential psychology and personality theories.

This is attempt to follow the basic principles of ontology **as the theory of objects and their ties (relations, dependences and predication).**

HSL specification contains phrases, object names, keywords focused on definition of any related to human-being object in terms of structure, dynamics, internal and external relations.

HSL NOTATION

<! ... >	<! *comment* >
<xxxxxx>	<! container e.g. <key-word> means key-words collection. >
<spec *specification-name* >	<! specification start - first line of specification>
</spec >	<! specification end >
<def *name* >	<! start of definition >
</def>	<! end of definition >
descr	**<!** tekstual description of element **>.**
layout	**<!** layout of element **>.**
:=	**<!** type assignment **>.**
xxxxxx := [<component1,...>**]**	**<!** [....]structure assignment of xxxxxx>
=	**<!** value assignment **>**
:	**<!** attribute,trait assignment **>**
::	**(., ., ., ..)**<! elements of list belong to>
= =	**<!** equivalency >
{.....}	**<!** delimiters >
(.,.,..)	**<!** list >
<x>+	<! can occur more than "x" >
/	**<!** or >
" '	**<!**delimiters of string & note>
(./ ./ ./ ./)	<! only one value / element of list may exist >
&	**<!** and >
XXXXX	**<!** ontological object >

YYYYYY.XXXXXX <! qualified name of Ontological object >

XXXXXX.type <! type of Ontological object >

name <! non-ontological ex. executive/operational object >

NAME(x) <! instance of ontological object >

name(x) <! instance of non-ontological OBJECT >

name:a <!attribute of object e.g. HUMAN:biological means: biological needs of HUMAN - see need :: (list)>

name[a,b,c,d] <! structure of object - simplified method of definition >

id = <! object identifier for a given instance of OBJECT e.g. Tom Smith, "id" may be written also as HUMAN="Tom Smith" or HUMAN(Tom Smith)>

id := [format id**]** <!structure of id:=[name, unique-code] >

OBJECT.type := **(OBJECT, elOBJECT, socialOBJECT)**<! type of ontological object>

elOBJECT :: **(HUMAN, ANIMAL, ...)** <! - elementary atomic object >

socialOBJECT :: **(COUPLE, socialGROUP :=(FRIENDS,WORKTEAM, PROFESSIONALS, NEIGHBOURS, NATION)**

object.type := (**i-object, v-object, d-object)**<! type of nonontological object >

i-object :: <! informational object e.g. birth statement >

v-object :: <! virtual object >

d-object :: **(event, action, process, behaviour)**<! dynamic object e.g. (action) >

Environment == ENV

ENV.scope := **(UNIVERSE, CONTINENT, COUNTRY, REGION, SITE)**

ENV.type := **(BIOPHYSICAL, GEOGR, CULTURAL, SOCIAL, LEGAL)**

ENV.LEGAL := <! ***Legal acts, resolutions, decisions etc.***>

ENV.CULTURAL := (tradition, history, education, religion, ideology, art, radio-tv)

ENV.BIOPHYSICAL := (Homosapiens, Animals ...)

ENV.GEOGR := **(<homeaddress>, COMPANY, SCHOOL)**
homeaddress :: (SITE,STREET,HOUSE,FLAT)

LIST OF KEYWORDS AND PHRASES

keywords :=(object,subject,type,scope,spec,def,descr,note,section, status,state,place,life-space, behaviour,process,action,event,trigger,drive,< relation > ...)

---

relation := (
activated by/when/if activates,
appearence depends on <! e.g. appearence of child object depends on existence of parent object >
assisted by,
belongs to, <! e.g. belongs to HOMO.HOMOSAPIENS>
owned by,
built of,
calls <object> (xxx,yyy) <! xxx input elements, yyy return elements >
consists of <parts> ,
contained in/contains,
controlled by / controls,
derived from,
included in ,
involved in,
driven by <! driven by date, driven by schedule,driven by frequency>,
existence depends on< <! condition of existance .... >
exists when/in/for,
evaluated as critical/mostwanted ,
linked to ...by / links,
matched/matches <! e.g. operation confirmation >
refers to <! direct relation > relates to, related by affinity, represented by /
represents ,
shared by / shares,
used by / uses/ not used/ misused/ abused
)

---

state := (active, inactive, dormant, suspended, aborted, idle, lost, dead, homeless, retired, married/divorced/single...)
place := (point, area, ...)<! -associated with a being there >
life-space := (psychological,social, educational,financial ...)<! -przestrzeń życiowa (in Polish), room-unlimited physically and associated with becoming, striving for newness and growth >
status := (generic, real, virtual)

subject <! subject of specification e.g. HUMAN >
event <! -elementary atomic fact >
action<! - complex activity >
process <!- sequence of actions and events with common aim, initiated by trigger, based on motivation >
process := [motivation/target, drive/trigger, action+, terminator]
behaviour <!- sequence of processes of the same ontological OBJECT >
behaviour.process :: [selfrealization, need, <actions>, satisfaction]
behaviour = <function>(life-space)
behaviour.serial :: [marriage, friendship,career, ilness, aging]
section : = (genotype, fenotype, olh, health, hobby....)
OLH -Object Life History := [birth, social_events, health_illness-events, educ-events, job-events, critical_events, death]
cluster <! Global Factor, Complextype - estimated on the base of several particular factors > **:= (**self, profile/type, attitude,leadership,ability,extraversion,anxiety,independence, healthState, creativePotential, happiness , lifeStyle, BipolarDimensionsofPersonality **)**
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, leadershipStyle, leadershipPotential**)**
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,self-actualization), financial, ...**)**

BipolarDimensionsofPersonality ::

(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), ...)

sex :: (male/female/x)
family :: (gentree, parent, child, son, doughter, grandsone, ...)
emotion :: (love, hate, satisfaction, frustration, agression, enjoyment, anger ...)
complex :: (fear of insupport, egression, inferiority,...)
habit :: (., ...)
hobby :: (., ...)
health :: (measures, physical-examinations, illness-history, ...) <! >
learning<!> ,
profession<! >,
role :: (drive, trigger, advisor, executor, generator, agent, component, integrator, caller, transmitter, tracer, spouse, manager, patron, partner, supervisor, participator, owner, stockholder, customer, supplier)
trigger/proactor <! trigger (of process or event) >,
reactor <! acceptance, rejection,constructor of environment for process >
generator <! generator of events >,
integrator <! integrator of complex object or set of objects>
tracer <! tracer of process >,
executor <! >,
participator <! participating object >,
component <! part of >
performance center <! >

dynamic characteristics(processes, operations. events etc.
mode of execution
(execution-mode := mode-rt <! real time-instant >,mode-bat<! batch-delayed-planned > mode-ond <! on demand >)
{control-flow <! characteristics of dOBJECT or "process" >
***repetition* := (iteration, circular mechanism, single spiral, multiband spiral** <! spiral pattern is used in learning processes and differs from iteration that each scroll could contains different mechanism and contents >)

activated .... by ...with <initial-value> at <time-point > when ..
finished at < > with <value> when ...}

{ info-signal flow
<info.> from (< operational object>)
to (<operational object>) thru path <! network points >
processed ***<algorithm name >***
}

{body :: (interface, contents) }
population layout := (free-space,swarm, network, hierachy, line, triangle, tunnel...)

Specification contains definitions of ontological and executive/operational objects related to a given human-being <def subject HUMAN(name)>. The base for such specification is metaspecification <def subject HUMAN>.

<HSLv.1> <! language indicator>

<spec(name)> <! first line of specification>

<def subject **(name)**> <! first line of definition>

a small example of specification

```
<HSL> <spec> (beta)
   <def subject HUMAN(Zygmunt Ryznar)>
      sex :: male
       family :: (married, parent of 3, grandfather of 4)
        state := (retired, active)
       status := real
       cluster.self = average
       ability :: (openMinded, creative, fast, abstracted )
       need :: psychological.self-actualization
       temperament :: (emotional,  sensitive,
                          introversive, tense, reserved)
       cluster.happiness = good+
```

NOTACJA JĘZYKA OSL

(po polsku - in Polish)

<! *komentarz*>

<! w nawiasach ostrych <> podawane są tylko głowne frazy strukturalne: def, spec oraz komentarze, nazwy obiektów i predefiniowane słowa kluczowe pisane są dużymi literami, obiekty niebiznesowe pisane są małymi literami>

<def *nazwa obiektu*> <!początek definicji obiektu>

</def> <!koniec definicji obiektu>

<spec *nazwa specyfikacji*> <!początek specyfikacji>

</spec > <!koniec specyfikacji>

::= <!przypisanie typu (np. klasy, obiektu) >

:= <!przypisanie do listy>

= <!przypisanie wartości czyli podstawienie >

: <!przypisanie obiektowi atrybutu>

:: <!przynależność>

= = <!ekwiwalentność>

{.....} <!ograniczniki frazy specyfikacyjnej>

(x,y,..) <!lista>

YYYYYY.UUUUU(XXXXXX) <!kwalifikowana nazwa obiektu >

[name] <!obiekt wykonawczo-operacyjny>

(XXXX)<!obiekt podstawowy>

keywords <!kluczowe słowa specyfikacyjne, są dziedziczone przez obiekty niższego rzędu>

Można założyć hipotetycznie, iż w przypadku komputerowej implementacji języka, pisanie specyfikacji byłoby wspomagane przez specjalizowany edytor (m.i. poprzez gotowe "formatki" ze zwrotami), generacje opisu poszczególnych obiektów byłyby utrzymywane w bibliotece specyfikacji, zaś specyfikacja całości tworzona będzie automatycznie w wyniku konsolidacji zmian.

Język OSL (Object Specification Language) składa się ze zwrotów standardowych, niezależnych od rodzaju biznesu, służących do definiowania obiektów oraz słów kluczowych specyfikowanych do używania w lokalnym obszarze (AREA) lub globalnie w ramach tzw. świata (UNIVERSE). Opis dokonywany jest w imieniu głównego podmiotu (SUBJECT). UNIVERSE, AREA i SUBJECT są więc jakby superklasami. Klasy w ramach AREA mogą być definiowane swobodnie (user-defined} zgodnie z potrzebami głównego podmiotu. W OSL wyróżnia się również tzw. obiekty wykonawczo-operacyjne, którymi są pracownicy podmiotu obsługujący klasy (np. lekarz naczelny,, lekarz chorego, itp.) oraz infrastruktura techniczna (w tym serwery, bazy danych itp.) i organizacyjna.

Środowisko (ENV) w jakim działa aplikacja opisywane jest następująco:

ENV:=(REGULATIONS, INFRASTRUCTURE)

ENV.REGULATIONS:=(*Akty prawne, regulaminy, zarządzenia*)

ENV.INFRASTRUCTURE:=(*ServeryDanych, SystemyOperacyjne, Transakcyjne BazyDanych, HurtownieDanych , SystemZarządzaniaSiecią, Struktura organizacyjna firmy)*

Specyfikacja obiektu obejmuje takie elementy jak:

-id <!identyfikator obiektu>

-infoWindow <!"okno na świat " - informacje uzewnętrzniane przez obiekt>

-dataTable <!tablica danych.>

-typ obiektu ObjectTypes : (eOBJECT<!obiekt elementarny np. klient>

-rola obiektu (ROLE)

-historia życia (OLH -Object Life History)

-relacje (RELATIONS)

-charakterystyka dynamiki(poprzez procesy, transakcje, zdarzenia; sposób ich inicjowania itp.)

-tryb wykonania (EXECUTION_MODE:= modeRT <!real time>,modeBAT<!batch> modeOND <!on demand>)

-przepływ danych (DATA_FLOW)

-przepływ sterowania (CONTROL_FLOW)

Poniżej wymienimy niektóre elementy (szczególnie te, których nie udało się użyć w przykładzie).

ROLE :=(commander <! obiekt sterujący procesem, odpowiedzialny>,

trigger <!obiekt inicjujący/uruchamiający proces, zdarzenie>,

generator <!obiekt generujący np. zdarzenia, cash flow>,

agent <!reprezentuje usługę np. agent w call-center >,

integrator <!obiekt integrujący obiekty>

monitor <!śledzi wykonanie/przebieg procesu>,

executor <! obiektwykonawczy>,

participator <!obiekt uczestniczący>,

stakeholder :=(owner, stockholder, customer, supplier; partner, employee)

component <!składnik>

performance center <!obiekt będący miejscem wykonania>)

RELATIONS:=(*(activated by / activates, activated when/if,*

appearence depends on <! np.pojawienie się dziecka zależy od istnienia rodziców>

assisted by, belongs to /is owned by , *built from* ,

calls <obiekt> (xxx,yyy) <!xxx elementy przekazywane., yyy elementy zwracane>

consists of <parts> , *contained in/contains, controlled by / controls,derived from,*

driven by OPERATION,driven by product ,driven by banking regulations,

driven by customer, driven by date, driven by schedule,driven by frequency

existence depends on <!np. istnienie obiektu zależy od .... >

exists when/in/for, evaluated as critical/most wanted , included in ,

linked to ...by / links, matched/matches <! np. uzgodnić operacje>

refers to <!bezpośrednie odniesienie> relates to, related by affinity, represented by /

represents , involved in, shared by / shares, used by / uses)

STATE := (active,inactive,dormant,suspended,aborted,idle, lost)

STATUS := (generic, real, virtual)

dOBJECT<! obiekt dynamiczny np. operacja>

iOBJECT<!obiekt informacyjny np. informacje o chorym>

vOBJECT<! obiekt wirtualny np. symulacja rozwoju choroby>.

dOBJECT ::= ZDARZENIE,OPERACJA,AKCJA,PROCES)

== (EVENT,OPERATION,ACTION,PROCESS) <!przykład równoważnego definiowania dwujęzycznego>

<!ZDARZENIE jest to elementarny niepodzielny fakt, czyn, działanie lub zaniechanie spodziewanego działania >

<!OPERACJA jest to sekwencja zdarzeń >

<!AKCJA jest to złożone działanie , składajace się z kilku operacji>

<!PROCES jest to ciąg akcji i zdarzeń posiadający wspólne zadanie inicjowany przez zdarzenie inicjujące i kończony przez zdarzenie końcowe) >

{CONTROL_FLOW

repetition :=(iteration, single spiral, multiband spiral <!spirala znajduje zastosowanie w procesie uczącym się i od iteracji różni się tym, iż każdy jej zwój posiada inne mechanizmy i treść >)

activated .... BY ...with <initial-value> AT <time-point > WHEN ..

finished AT < > with <value> WHEN ...}

{ DATA FLOW

<.> from (<.>) <!lista danych przychodzących od...>

<.> to (<.>) <!lista danych wysyłanych do>

<> dataGeneration Method <*nazwa*>

<>dataEncoding Method <*nazwa*>

<>dataCompressing Method <*nazwa*>

}

{BODY ::= (Contents, Script, Metadata)

contents <!zawartość ciała np. treść dokumentu, kod programu>

population layout := (free-space,swarm, network, hierachy, line, triangle, tunnel...)

script <!lista operacji generowanych wewnątrz obiektu odpowiednio do jego zachowania się np. oczekiwanie na dostęp do danych, zawieszenie się>

metadata <! np. w XML- opis struktury ciała obiektu >}