DOCUMENTS FOR INTELLIGENT AGENTS IN ENGLISH

Sandor M Veres and Levente Molnar
School of Engineering Sciences
University of Southampton
Highfield, Southampton, SO17 1BJ, UK.
Email: s.m.veres@soton.ac.uk

ABSTRACT
The paper presents an information processing system for autonomously operating vehicles where engineers can write ”publications” written in English that the autonomous systems can ”read” to acquire knowledge such as various skills and behaviour policies. Knowledge about how to perform feedback control based operations, how to do dynamical modelling, path planning, servo and tracking control skills, vision based feedback control, etc. can also be transferred. The ”publications” are similar to engineering booklets with contents, sections, subsections in English, that can appear in HTML, LaTex(pdf) formats. The same paper's HTML file can be read by an agent on board the autonomous vehicle and after reading the paper the agent knows how to alter its control of the vehicle. There is no need for an application engineer to read a journal publication and implement it by programming, the research engineer's methodological work is directly utilized by the autonomous vehicle or robot. Engineers different from the author of the ”publication” can also read the papers and learn the details of how the vehicle operates, how decisions are reached and how skills are performed. Not only will users of the autonomous vehicle clearly understand how it operates but will also know its limitations to avoid misuse or misunderstanding. Users can modify English sentences in the ”publication” to influence the vehicle's behaviour or how its skills are performed.

KEY WORDS
Autonomous vehicles, world modelling, map interpretation, collision avoidance, natural language programming, artificial intelligence, publishing knowledge for autonomous systems.

1 Introduction

The process of knowledge distribution among scientists and engineers is by way of publishing at conferences, in journals and books. They describe in technical papers how a control method works, how a navigation method proposed by an author works. Also information processing and decision making procedures are published by research engineers (REs) [1]. Other engineers, let's call them application development engineers (ADEs), read these technical papers, conceptualize the meanings and may opt to implement the methods in practice. There are thousands of control and signal processing engineering papers written every year worldwide. Their impact on industry is patchy as many of the papers are not read or chosen to be ignored due to financial necessities of companies. Knowledge transfer to industry (KTI) is slow as is well known. Cynics may even ask: ”why do we need to speed up KTI?... we are fine as we are”. Others may say: ”this is a topic for our knowledge engineers ...but artificial intelligence is far from being able to help yet.”

Reference [2] describes principles and reviews systems developed to support software developers, that also act as mediating tools to provide a platform facilitating knowledge sharing. The need of supporting knowledge collaboration in software development environments, is highlighted in [2], based on the conceptualization of software development as knowledge-intensive and distributed cognitive activity.

This principle is extended in [3] for the context of shared knowledge and collaboration between human users, i.e. scientists, engineers, and machines or intelligent agents. It is proposed that documents written in English are distributed to networked devices that human users can also read and hence develop shared ”understanding” with the devices. Hence a system is created that allows humans and machines to communicate intuitively (i.e., by using human language) through ”intelligent cooperation”. This is achieved by creating a coherent meaning-system understandable by both the devices and people.

Is it however possible to make a machine to ”read” a technical book written in English sentences to formulate methodologies at the conceptual level? Some would say: ”natural language interfaces have been developed and you can use various controlled English languages. These need a priori knowledge and definition of meanings .. and you need to know the grammar and what phrases to use .... the problem is a complexity nightmare!”

And yet this paper is describing an existing system whereby engineers can write documents in English that can also contain equations, figures and images, quotes, numbers, physical quantities, etc. that suitable intelligent agents can read and apply it in their daily work [1]. This paper describes a system with the following practical features:

(1) The authors and readers of these machine readable documents (called ”system English” papers, for short

called sEnglish papers) do not need to learn grammar, just to apply common sense. The most important thing for an author is to have conceptual clarity in their area and express that in a suitable style: ”writing skills are needed”. It only takes a few hours to be trained as an author.

(2) The autonomous systems can read these special engineering publications in sEnglish. sEnglish publications are of special format with contents, sections and subsections. They may contain images, mathematical formulae to support understanding both to humans and to autonomous systems who read them.

(3) The autonomous systems who read these documents do not need access to a central dictionary or library of meanings. There is no need for building up an infrastructure so that autonomous systems can read sEnglish books. The system can be used immediately, without any investment. Autonomous vehicles (AUVs, UAVs, AGVs and spacecraft), robots, toys can be built that understand natural language documents without a supporting infrastructure to be set up first.

The question could be asked that: ”do we need publications for autonomous systems that humans can also understand?”. The right question to ask is rather ”do intelligent autonomous systems need to read publications that humans can also read?”. We argue in this paper that the answer is an emphatic yes. This is not just a ”nice feature”. Machines reading publications is going to become a necessity if we want truly intelligent autonomous systems in the future.

The reason for ”publication for autonomous systems” is needed is complex and the two most important factors are:

(1) We need to pass on knowledge to our ”servant robots” and we need to be aware what they know and how they know it, as that can help to avoid misunderstanding.

(2) The development of mental and physical skills of intelligent autonomous systems (by humans) is much faster than how fast we can economically afford a capable hardware to be replaced. Hence we need ”software upgrades” - by the authonomous systems reading our books and papers.

We argue that software downloads for intelligent autonomous systems are old-fashioned and outdated now - or will be in the near future. Autonomous systems and robots will read books on mental and physical skills and facts about the world from documents that their user can also read with ease and hence they can have a shared understanding of a selected part of the world.

In this publication system the manufacturer of a vehicle platform or robot can invite publications from research engineers to enrich the knowledge base of agents controlling these platforms in a format that also merits current

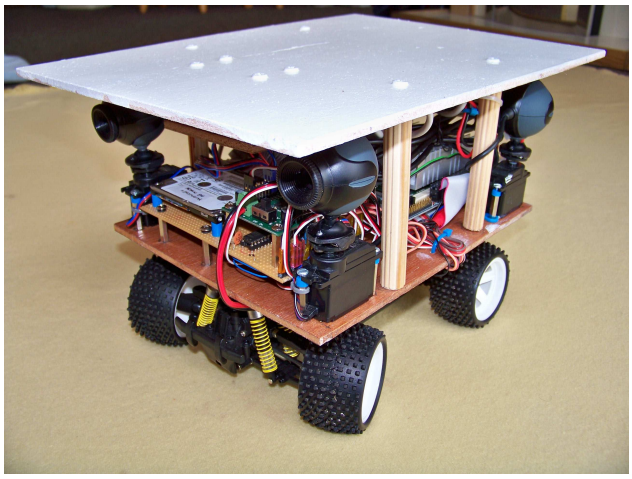

Figure 1. The autonomous ground vehicle platform used for the demonstration.

standards of scientific and engineering publications. This could for instance be the case for the use of autonomous vehicles, robotic pets and intelligent toys, gardening robots or agricultural unmanned autonomous aerial vehicles.

Publications for autonomous systems is a logical next step of information systems in a historical perspective [3]:

(a) There was first verbal communication between people.

(b) Writing was developed to record events and knowledge.

(c) Later book was invented, i.e. multiple copies of written material were printed that masses of people could read.

(d) Journal and newspaper publishing and the WWW exists to distribute knowledge.

In the computer science community the idea of ”natural language programming” has been largely considered impossible and impractical due to ambiguity. This paper reports about a complete system for autonomous systems unambiguously understanding human readable books to enhance their skills and knowledge about the world.

There is an sEnglish (short for system English) authoring Tool (sEAT) and sEnglish reader agent system (sERA) [4] that does the job described above. A simple reactive agent (sERA) that ”understands” these papers and can run on embedded computers or on a PC [4] is available. The system is well tested and has been formally verified [4]. This paper describes this system of ”publications for autonomous systems”. The methodology is illustrated on the control of an experimental autonomous ground vehicle shown in Fig. 1.

In this paper the main contributions are as follows:

- Laying down the theoretical foundations of natural language programming (NLP) in Section 2 by formal definitions.
- Explaining the practical usefulness of NLP in the creative abstraction process of software engineering and knowledge sharing among programmers. (Section 3)
- Introducing the concept of publications for intelligent agents (Section 4)
- Illustrating the use of NLP in sEnghlish document by a Jason based BDI (beliefd desires intentions) agent. (Section 5)
- Illustrating the use of NLP in sEnglish on programming of an AGV (autonomous ground vehicle). (Section 6)
- Comparison with controlled english (Section 7).

2 Theoretical foundations

An overview of the essential attributes of ontologies for use by multi-agent systems in distributed computing is given in [5]. It is stated that ontologies are needed for domain knowledge representation in order to meaningfully support agent inter-operation. Contributive factors to the development and study of ontologies for knowledge sharing and reuse are reviewed in [6]. The following definition is generally accepted by the research community [5]: "Ontologies are explicit formal specification of a shared conceptualization" [7], [8], where:

- *explicit* means that "the type of concepts used, and the constraints on their use are explicitly defined" [9];
- *formal* means that "the ontology should be machine readable, which excludes natural language" [9];
- *shared* "reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group" [9];
- *conceptualization* emphasizes the abstract model of some phenomenon in the world by having identified the relevant concept of that phenomenon" [9]. Conceptualization refers to the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [10]; i.e. addresses the abstract, simplified view of the world that is required for representation [7]. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly [7].

Ontologies enable content specific agreements to facilitate knowledge sharing and reuse among systems that submit to the same ontology/ontologies by the means of ontological commitments [5], [11]. Ontological commitments are desired to be specified at the knowledge level [7], [12]. An agent commits to a knowledge-level specification if its observable actions are logically consistent with the specification [7]. Ontologies describe concepts and relations assumed to be always true in a particular domain by a community of humans and/or agents that commit to that view of the world [5], [9]. It has also been pointed out that ontologies can be machine interpreted and, if not directly human-readable, they should at least contain plain text notices or explanations of concepts and relations for the human user [5], [13], [14], [15]. In ontology design much attention needs to be dedicated to aspects of human readability [16]. For clarity and coherence natural language documentation and examples, in addition to the axiomatic parts, should be used [16]. The degree of formality by which the vocabulary of an ontology is specified can vary from informal definitions expressed in natural languages to definitions stated in formal languages such as first-order logic with a rigorously defined syntax and semantics. Current use of ontologies ranges from glossary to formal requirements such a interoperability among software tools [16].

First a formal definition is provided the shared knowledge base ontology of all agents including the humans.

Definition 2.1 *An agent's modelling ontology* $\mathrm{O}{=}\langle\Gamma\,|{\prec}|\,@|\Lambda\rangle$ *consists of a lattice* $\langle\Gamma\,|{\prec}\rangle$ *over the class set* Γ*, an attribute label set* Λ *and an attribute mapping* $@:\Gamma\rightarrow 2^{\Gamma(\Lambda)}$ *where* $\Gamma\,(\Lambda):\Gamma$ *is a mapping from the attribute label set to the class set* Γ *. The class set has a universal sup class* ξ_0 *that is the modelling object so that* $\forall\xi\in\Gamma{:}\ \xi\ \prec\ \xi_0$ *and a universal sub model class* ξ_∞ *such that* $\forall\xi\in\Gamma{:}\ \xi_\infty\ \prec\ \xi$ *. The* $@$ *is also required to satisfy the inheritance condition* $\forall\xi,\zeta\in\Gamma{:}\ \xi\prec\zeta\ \Rightarrow\ @\,(\xi)\supseteq@\,(\zeta)$ *.*

Human agents play a special role whose knowledge is not fully formalised but partially described by the following definition of conceptual graphs.

Definition 2.2 *A conceptual graphs* $G=(C,R,G_e,O_c,O_r)$ *is a directed bigraph whose vertices are decomposed into two sets of concepts and conceptual relations* $G_v=C\cup R$ *and their edges are defined by a subset* $G_e\subset C\times R\times C$*. Both* C *and* R *are concepts from some ontologies* O_c *and* O_r *respectively.*

Human knowledge can be modelled by sets of conceptual graphs (CGs) that can also have modalities of past, present and future or indirect speech, etc., for instance $pasthour(G_3)$ or $communication(Peter,G)$. A CG can also be an abstraction of a more complex conceptual model described by graphs. Hence CGs can also be related by "abstraction of": $G_1\triangleleft G_2$ if G_1 is and abstraction of G_2. Some conceptual graphs are based on perception of the surrounding physical world and are direct abstractions from perception processes. We will not formalise human thought processes but make the following assumption.

Assumption 2.1 *Human knowledge* $H_{KB}=\{G_i\mid i\in J\}$ *about the present and the past of the world is represented by a time varying, non-unique set of conceptual graphs*

with temporal, communications, perception and abstraction modalities.

In the following a text is understood as a sequence of ASCII characters. A word is a sequence of characters without space. A proper name is a word starting with a capital letter but not all words starting with capital letters are proper names.

Definition 2.3 *Let N be a computer programming language with syntactically correct code set L_N and let N_l denote a natural language. A natural language program is a tuple $N_p = \{O, S\}$ of and ontology O as defined above and a set of natural language sentences S. Any $s \in S$ has the following properties:*

1. *s is a sequence of words that starts with an initial capital and ends with ., ! or ?.*

2. *Some subsequences of words in s can correspond to either property names or class names of O and some words can be proper names representing individuals of classes in O.*

Each $s \in S$ has a meaning $m(s)$ that is either a $m(s) \in L_N$ or is a set of ordered sentences $m(s) = \{s_1,\ s_2,\ ...\ s_k\}$. A compilation or translation of S into N is a mapping $T : S \rightarrow L_N$ that is consistent, i.e. $T(m(s)) = concatenation(T(s_1),\ T(s_2),\ ...\ T(s_k))$. The N_p is called a natural language program *if there is a unique T such that $T(s) \in L_N, \forall s \in S$ and all $s \in S$ are acceptable in N_l.*

The relationship of a conceptual program to human thinking, and to knowledge sharing among a set of programmers, is formulated by its interpretation.

Definition 2.4 *A pair of functions $M = [M_r, M_c]$, $M_r : S \rightarrow H_{KB}$ and $M_c : \Gamma \rightarrow C$ is called the* interpretation *of a natural language program $N_p = \{O, S\}, O = (\Gamma, \prec, @, \Lambda)$.*

The usefulness of natural language programs (NLPs) stems from the fact that natural langauge sentences are (by their very anture) suitable for shared interpretations between humans. As humans have shared interpretations of sentence meanings in their brains, NLP provides shared interpretation of meaning in terms of a computer program.

Theorem 2.1 *Any NLP provides shared interpretation of sentence meaning between human and a software agents that is able to execute the translated code of sentences.*

Proof. Any sentence s in an NLP N_p corresponds to a code $T(s)$ that is its meaning for an agent residing on a computer. On the other hand any sentence s in an NLP also has an intepretation $M_r(s) \in H_{KB}$. The correspondance $T(s) \Leftrightarrow M_r(s)$ creates a shared meaning between the software agent and the human. Q.E.D.

3 NLP for knowledge sharing

There are two main contributions that NLP makes to software engineering:

- Supports the creative coneptual process of software creation in the iterative process of ontological concept and sentence definitons.

- It provides a hierarchy of meaning definitions that programmers can easily share while the meanings compile into code unambiguously.

In a sense these two features are complementary: one is addressing developing deepening knowledge and the other one is addressing sharing this knowledge.

Reference [19] describes the hierachical abstraction layers of programming and points out that block diagrem based abstractions of programs are ill placed, they are two low resolution. Eventually the author advocates the development of "Developmental Mathematics" which can be a complex process. In this paper we neetly fit NLP into the abstractions layers of programming. As shown above NLP provides simultaneuous human (through the use of natural language) as well as machine interpretation (translated code) of an NLP sentence. Note code unambiguity preserves the cornerstone of digital computing: the usefulness of digital computation is its detrminism, i.e. its reliability in banking, accounting and computer control of aircraft, machines and robots, etc.

Reference [2] describes the need for knowledge sharing among engineers developing large software systems. NLP provides a solution for the following reasons:

1. The ontology of the NLP can be based on natural language professional terms for classes and properties that all professionals understand.

2. The sentences representing code are formulated in terms of share human professional concepts.

This means that colleagues can read into each other's code at any part of their work and can interpret the meanings and structure of the other's program.

4 Publications for agents

The process of publishing for autonomous systems can be split into three parts: (1) authoring papers, (2) distributing them in HTML format for web browsers and as PDF documents, (3) making agents to read these papers and let them use their content in their operations.

4.1 Authoring papers

The process of publishing for both agents and fellow engineers, is displayed in Fig.2. There is an authoring tool to write a paper in English sentences where the meaning of

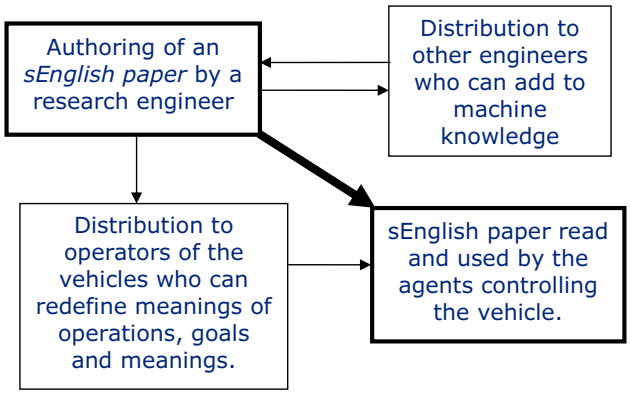

Figure 2. The principle of authoring and distributing natural language programs that are presented as sEnglish papers.

sentences is explained by other sentences until the meaning reaches signal processing level and no more conceptual meanings need to be defined: at this level MATLAB code or similar high level code (Octave, SciLab or Phyton) is inserted that represents ”unconscientious” operations. Goals, actions and modelling statements about the world can all be expressed in natural language sentences. Fig. 3 displays a window of the authoring tool sEAT. The start of an example is presented in Fig. 4 as a web document in HTML. The high level code can be interpreted by the agent for its realtime system (e.g. TTTech or LonWorks)

The abstract of the paper describes the conceptual relations to other areas of knowledge and where the contribution of the paper lies and in particular it refers to the type of hardware system where the agent operates.

An sEnglish paper starts with an ”Abstract”, ”Conceptual structures” used and continues with sections and subsections that describe the meanings of activities. An activity can be represented by several sentence formats meaning the same. The actual meaning of a sentence can be always explained by other sentences that may be explained by further sentences or they refer to a piece of computer code.

5 Distribution of papers

sEnglish papers can be either in (1) HTML formats for direct transfer to agents through a local network or the Internet or in (2) PDF formats for mainly human reading of the papers. Figures 4 and 5 show the beginning of an sEnglish paper in HTML. The contents of the paper in Fig. 5 is not for human consumption only, the agent reads the concepts and activities from the contents and the rest of the text of the paper.

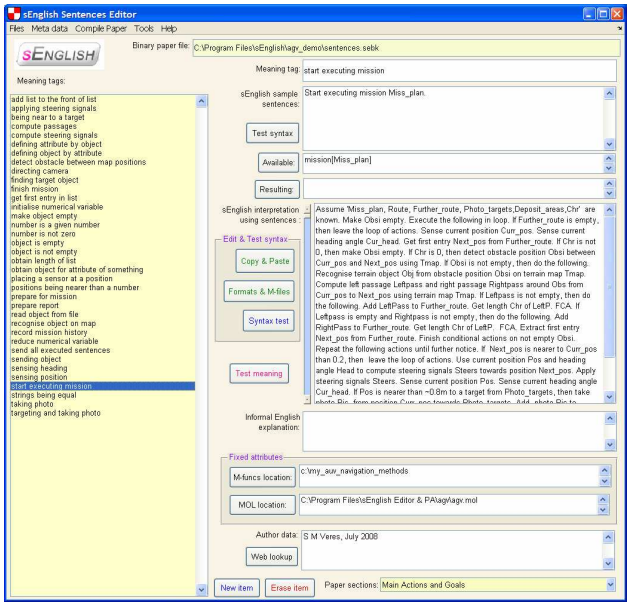

Figure 3. One of the authoring tool GUIs to type in meanings of sentences by other sentences.

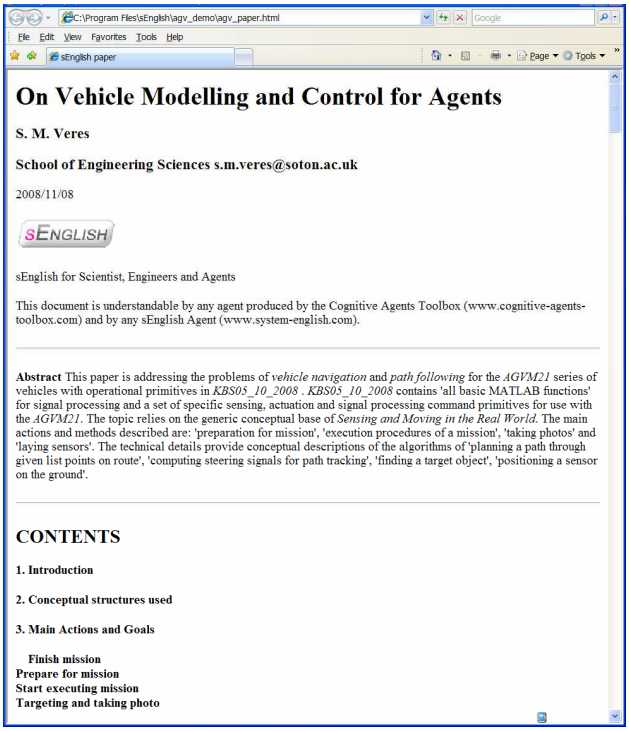

Figure 4. The initial part of a natural language program (NLP) written in sEnglish and represented as a web page (HTML document).

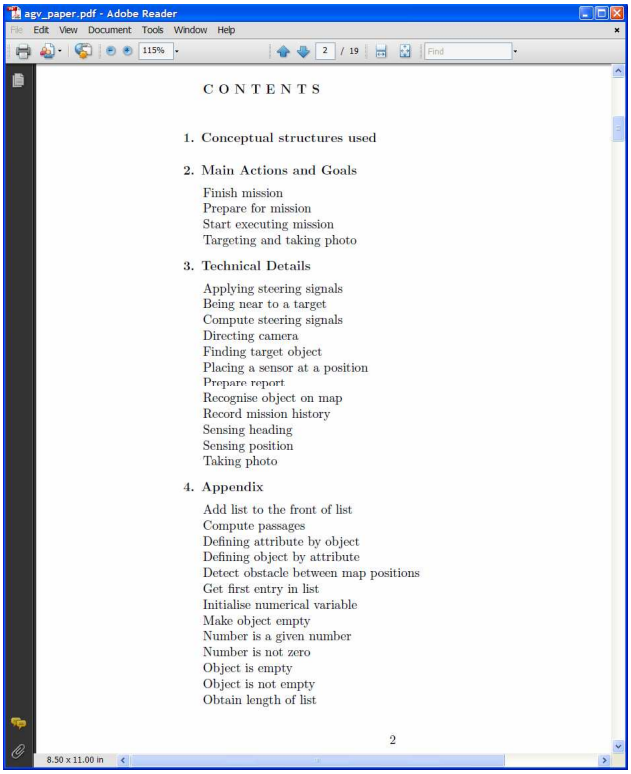

Figure 5. The contents part of the same natural language program (NLP) written in sEnglish and represented as a PDF document (HTML document).

5.1 sEnglish based reactive ”behavioural” agent

Reactive agents are simple and can be based on finite state machine definitions. There are also layered agent architectures that evolve abstractions from low level sensors to symbolic computation and then from decision making back to coordination and planning in detail to low level.

A simple agent that has finite-state-machine transitions prescribed by exit conditions and entry conditions (such as those defined in hybrid systems or in StateFlow^{TM}), can be defined by the use of sEnglish sentences [5]. This agent is called reactive or situated as the environment and internal events determine the course of action the agent takes. There is no deliberation of intentions and plan selection or plan building. Reference [5] provides a methodology of how reactive agents can be defined using an sEnglish document. The agent defined in sEnglish can be compiled into StateFlow^{TM} for simulation and into ISPL (interpreted system programming language) for formal verification by model checking.

Another simple agent, that can execute reactive behaviour, as defined in an sEnglish paper provided to the agent, is available under the name sEnglish Reader Agent (sERA) in association with the sEnglish Authoring Tool [4]. This agent can read sEnglish publications in HTML format and use them to control an embedded system or a PC based control system. Its user interface is displayed in

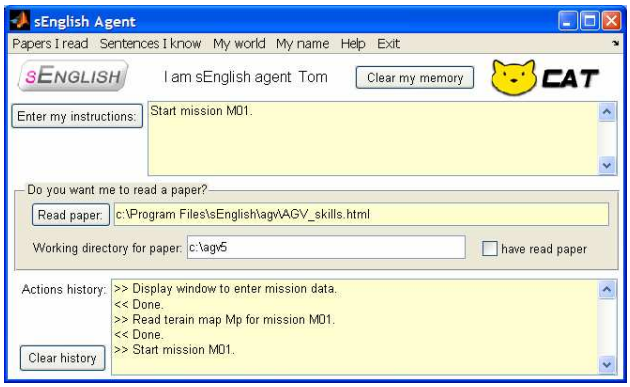

Figure 6. The user interface of the agent that can read and interpret technical papers in sEnglish, to carry out tasks and achieve goals. Sentences used can be utilised for both execution as well as for communication.

Fig. 6. sERA’s main functionality includes:

- Read an sEnglish paper in HTML file from a computer memory device, from a local network location or from a URL as instructed via the GUI of the agent.
- Execute the meaning of a sentence entered via the GUI dialogue field of the agent
- Receive a message over the network that asks the agent to read an sEnglish paper from a depository on the Internet.
- Execute the meaning of a sentence sent to it via network such as ”Start mission M01.”.
- Display history of a dialogue between itself and other agents, including any human contact and communications.
- Perform messaging, sensing and control operations as prescribed by sentences with meanings as defined in the sEnglish paper read by them.

Though simple, sERA based agents easily lend themselves to formal verification of behaviour. The sEnglish sentences define abstractions of actions, environmental events and sensing/modelling of the environment. For easy understanding of operations for human operators, the state transitions can also formulated by ”If... , then ... .” sentences.

5.2 Advanced, deliberative BDI agents

Belief-desire-intention (BDI) agents can be made capable to read and use sEnglish papers with small (re)configuration effort. We illustrate this on the Java based *Jason* agents.

Agent based control of autonomous vehicles or robots is the only proper way of writing software. Assume someone ignores this statement and writes a code for a vehicle or robot using a hybrid system specification as a finite state machine with continuous flows within states (for instance in StateFlow^{TM}/MATLAB/Simulink by MathWorks Inc.). All that this engineer is going to create is essentially a reactive agent. Agent research has however shown that more sophisticated agents can be more adaptable and can have problem solving capabilities in real autonomous missions where advance specification of how to respond to all possible events of the environment is impossible or uneconomical to preprogram. Hence doing autonomous control via finite-state-machine definitions is viable but is a subset of more sophisticated agent based approaches.

One of the most sophisticated deliberative architectures are the belief-desire-intention (BDI) agents. These agents maintain a belief data base B that is regularly updated using sensors and signal processing and also process incoming communications events. A list of goals are maintained by the agent that contains the goals set by the human operator of the agent, nonetheless, the agent can extend this list by temporary goals, that arise from its planning to achieve top level goals. Some events (that can be internal or external) and all goals are associated with action sequences, that are called plans to achieve the former. In some common and fast executing BDI languages (such as AgentSpeak, Jason, Jack or Jade) all plans are to be declared by the agent programmer. Although this makes the agent less "creative" than logic inference based rational agents, as the plan set is fixed and not generated by the agent, they have the great advantage of fast execution and easier formal verification. In many safety critical systems such formal verification of a sophisticated BDI agent is crucial. Hence these kind of BDI agents combine the advantages of deliberative agents with the advantages of reliability that is fundamental for industry.

The following is part of the code of a BDI agent written in Jason that does deliberation in terms of having a set of goals, that selects the most suitable intention for short term action, and a plan that it executes before carrying on. It is about a GEO (geostationary earth orbit) satellite's control that keeps its position in term of latitude and altitude so that it can carry our communication functionality. The advantage of BDI agents in this context is that the agent can resolve complex problems of malfunction on board, to serve its ultimate purpose of providing television and other broadcasting. The "// sE:" lines show sEnglish sentences that are explained in the sEnglish publication and are presented by a comment line of the corresponding Jason external call.

```
//===================================
// Jason/sEnglish SATELLITE CONTROL
// 25.4.2009, SMV
//===================================
// See the sEnglish document geo-pap.html
```

2 Control Planning Procedures

2.1 Preparing plan to go to a location

Sentences to use:

Prepare force sequence P to go to location Loc2 .

Available things are: Loc2(location) .

Details of the meaning:

This is a general procedure to obtain a sequence of pulse forces that moves the satellite from one location to another . The exact code goes as follows . Let P be a 'force sequence' . Recall control configuration C from memory . Let M be the 'force-to-thrusters conversion matrix' of C . Let B the 'thruster bounds' of C . Optimise fuel optimal force sequence P for thruster bounds B , matrix M to go from Loc to Loc2 using sampling period 3s . The action of 'preparing plan to go to a location' links up with jason statement 'cont.plan_approach' .

Resulting things are: P(force sequence) .

2.2 Preparing plan to go to centre

Sentences to use:

Prepare force sequence P to go to centre .

Details of the meaning:

This is a special case of getting a sequence of pulse forces to move the satellite from one location to another , namely the centre of the target box in this case . Let Loc0 be a 'location' . Set the 'vector' of Loc0 to "0,0,0" . Prepare force sequence P to go to location Loc2 . The action of 'preparing plan to go to centre' links up with jason statement 'cont.plan_approach_to_centre' .

Resulting things are: P(force sequence) .

Figure 7. Part of an sEnglish document that declares usability of the sentences in a Jason agent's logic.

```
// PLANS
@homing !get_to_centre :
not_in_centre & at(Loc) <- 
cont.plan_approach_to_centre(P,Loc) ;
!try_execute(P).
// cont.plan_approach -
// sE: Prepare force sequence P to go to location Loc2 .
// cont.plan_approach_to_centre -
// sE: Prepare force sequence P to go to centre .
(does not require naming the centre)
@exec !try_execute(P) : not_in_centre <- 
?control_hw_changed;
!reconfigure_control_hw(C);
cont.apply_controls(P,C).
// cont.apply_controls(P,C)
// sE: Apply control control force sequence P
using control configuration C with feedback regulation.
@control_hw !reconfigure_control_hw(C) :
control_hw_changed <- 
cont.get_actuator_data(D);
cont.compute_control_configuration(C,D).
// cont.get_actuator_data(D) -
// sE: Get actuator data D from memory.
// cont.compute_control_configuration(C,D) -
// sE: Compute new control configuration C using actuator data D.
@emergency !send_sos : total_hardware_failure <- 
cont.get_actuator_data(D);
com.send("MissCentre","all failed",D).
```

// com.send(”M”,”all failed”,D). -

// sE: Send to ’M’ message text ’all failed’ and actuator data D.

@totalfail +total_hardware_failure : true < −

get_thruster_rw_data(D);

cont.there_is_no_feasible_reconfiguration(D).

// comp.there_is_no_feasible_reconfiguration(D) :

// sE: There is no feasible reconfiguration for actuator data D.

//BELIEF UPDATES

@pos +at(Loc) : true < −

comp.distance(Loc) & D>2;

+not_in_centre.

// comp.distance

// sE: Compute the distance D of Loc from the centre.

The above Jason code illustrates in comments how the sEnglish paper with its sections links up with the logic of the agent code. Each ”cont.action_name” or ”com.messaging” external call in this Jason code has a corresponding subsection in the sEnglish document that unambiguously compiles into computer code. The agent reads the sEnglish document and its knowledge about control skills, world modelling skills and its behaviour constraints are updated in a way that it will have a well defined *shared understanding* with the operator-engineer who will also read the sEnglish document. This makes day to day practical work with the autonomous system not only safe but enjoyable for the engineers.

6 Example: The mission execution of an AGV

A webpage section that displays some AGV operations for ”Preparing for mission” and ”Executing the mission” is displayed in Fig. 8. The full sEnglish paper can be found at http://system-english.com [4]. The sentences used are concerned with mission data such as the route taken, plan of mission, list of photo targets, locations where sensors need to be deposited, etc. Also sensing of the environment and position is expressed in terms of sentences. Obstacle detection checks whether there is an obstacle on the straight line between the current position and the next waypoint. If there is an obstacle ahead of the vehicle, then a path is worked out either to the left or right from the obstruction.

6.1 Description of the AGV movements problem

Fig. 9 displays the laboratory environment where the AGV needs to move around. The model houses are located near to each other. At some places a passage is only twice or three times the width of the AGV. On the other hand the AGV is not able to turn on a smaller radius than 3 times its width. Hence its steering is seriously limited. As a mission continues, the battery of the AGV provides less and less power that affects the speed and angle of turning of the vehicle while the same control signal is used from the embedded computer. This of course necessitates the use

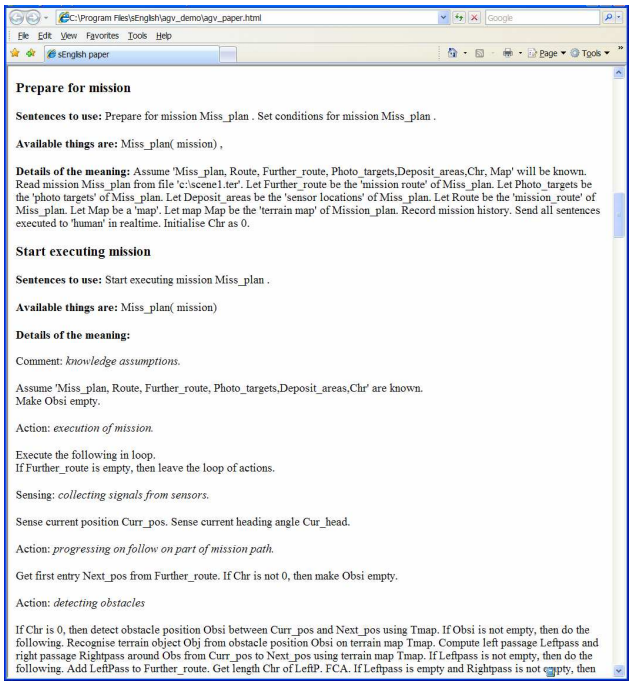

Figure 8. Start of two sections on ”Prepare for mission” and ”Start executing mission” .

of feedback, that in this example is put into vehicle operations in sentences. The AGV uses camera based navigation system. The AGV has four onboard cameras with servo controls of their viewing directions. In this demo also an external overhead camera was used to detect the location of the AGV within the environmental model and relayed back to the vehicle via a wireless network communication. Heading angle of the vehicle is estimated from past sensed positions and steering angle records. Inertial measurement units (IMUs) can be easily integrated to enhance the precision and reliability of vehicles navigation. All data fusion and world modelling is ”programmed” in sentences.

6.2 Path planning and execution skills

There is a considerable amount of literature available on path tracking of autonomous or semi-autonomous vehicles. Although [18] deals with environmental constraints, direct application of this scheme was not possible as the future path of the vehicle is seriously limited by its current heading. What matters is that passing through gaps can only be done by starting from a bounded locality. This necessitates more careful planning and an arbitrary waypoint sequence among the houses is not executable at all. The sEnglish paper of the AGV shown in Fig. 9, is detailed at http://system-english.com, and contains sections that describes the path execution of the AGV in English sentences, that is easy to understand and debug. Causes of possible problems in the course of a mission also become clear to the operators,

Figure 9. The environment where the AGV (lid removed in this picture) needs to navigate and pass through among houses.

much more so than that is possible with ordinary programming, that separates meaning from what the program does.

7 Comparison of sEnglish with other controlled English texts

NLP (sEnglish) sentences compile into a high level language to form a ”meaning” and not into predicate representations as in Attempto Controlled English (ACE) [20]. ACE is a language intended for formal world model descriptions and communications, it would be difficult to write computer programs and procedures in ACE. On the other hand NLP can be used to described modelling relations if the meaning is defined as a code changing a ”world model” of the agent.

These differences of essential importance can make ACE a complementary system NLP of knowledge representation to build a joint powerful systems where underlying functionality is defined in NLP. Procedures and communication procedures of engineering systems can be addressed more effectively by such a system, with an emphasis on autonomous systems or distributed engineering projects employing large teams of human developers.

John Sowa’s Common Logic Controlled English (CLCE) [21], is a logic traslation of English sentences. Under certain conditions the translation can be bidirectional from first order logic (FOL). CLCE especially suitable to exepress mathematical relationships precisely in English.

Processable English [22], in a similar way to Attempto and CLCE, translates a syntactically correct English text into first order logic (FOL) formulae.

8 Conclusions

This paper describes a ”publishing” system for autonomous systems where the documents are written in English and the vehicles can read them to improve their navigation, sensing and control skills with regards to self-movement and manipulation of external objects. They can also read behaviour rules and limitations. The advantage of using a publishing system over traditional ”reprogramming” are

- The human users will share the knowledge of the agents and that reduces misunderstanding when complex intelligent behaviour is needed during autonomous missions.
- The ”published” papers can be distributed to agents of a specific control program that define their deliberative or reactive behaviour. Instead of reprogramming, the autonomous systems learn from publications as humans do.
- The sEnglish publications can be distributed within a company or on the Internet as ”proper publications” to make colleagues aware of results in sensory signal processing, navigation, environment modelling and adaptive/learning control methods.

This paper illustrated the system on a small autonomous ground vehicle. The system can also be used with AUVs, autonomous UAVs, spacecraft and in most autonomous robots. For further background reading the reader is referred to [3].

9 References

[1] S.M. Veres, Natural language programming of semi-autonomous and embedded system?, Autonomous Systems SIG Articles, Electronics Knowledge Transfer Network (EKTN), UK, *http://www.electronics-ktn.com*, 2009.

[2] Y. Ye, Supporting Software Development as Knowledge-Intensive and Collaborative Activity, *Proc. of 2006 Int. Workshop on Interdisciplinary Software Engineering Research*, Shanghai, China, 2006.

[3] S.M. Veres, *Natural Language Programming of Agents and Robotic Devices* (London: SysBrain Ltd., 2008), ISBN 978-0-95584417-0-5.

[4] sEnglish Reader Agent and Authoring Tools, *http://system-english.com*, SysBrain Ltd., 2009, London, UK, 3 More London Riverside, SE1 2RE.

[5] C. Chira, A Multi-Agent Approach to Distributed Computing, *Computational Intelligence Report No. 042007*, Computational Intelligence Research Group, Babes-Bolyai University, Cluj-Napoca, Romania, 2007.

[6] V.O. Chira, Towards a Machine Enabled Semantic Framework for Distributed Engineering Design, *PhD Thesis at Galway-Mayo Institute of Technology*, Galway, Ireland, 2004.

[7] T.R. Gruber, A Translation Approach to Portable Ontology Specifications, *Knowledge Systems Laboratory Technical Report KSL 92-71*, Computer Science Department, Stanford University, Stanford, California, 1993.

[8] W.N. Borst, Construction of Engineering Ontologies for Knowledge Sharing and Reuse, *PhD Thesis at University of Twente, Enschede*, Twente Enschede, The Netherlands, 1997.
[9] R. Studer, V.R. Benjamins and D. Fensel, Knowledge Engineering: Principles and methods, *Data and Knowledge Engineering*, 25, 1998, 161-197.
[10] M.R. Genesereth and N.J. Nilsson, *The Logical Foundations of Artificial Intelligence* (San Mateo, CA: Morgan Kaufmann Publishers, 1987), ISBN 978-0934613316.
[11] P. Spyns, R. Meersman and M. Jarrar, Data modelling versus ontology engineering, *ACM SIGMOD Record, Special Issue: Special section on semantic web and data management*, 31(4), 2002, 12-17.
[12] A. Newell, The Knowledge Level, *Artificial Intelligence*, 18(1), 1982, 87-127.
[13] N. Guarino, Formal Ontology and Information Systems, *in Formal Ontology in Information Systems, FOIS'98: IOS Press*, Trento, Italy, 1998.
[14] M. Uschold, Knowledge level modelling: concepts and terminology, *Knowledge Engineering Review*, 13(1), 1998, 5-29.
[15] P. Borst, H. Akkermans and J. Top, Engineering ontologies, *Int. J. Human-Computer Studies*, 46(2-3), 1997, 365-406.
[16] M. Uschold and M. Gruninger, Ontologies: Principles, methods and applications, *Knowledge Engineering Review*, 11, 1996, 93-136.
[17] L. Molnar and S.M. Veres, System Verification of Autonomous Underwater Vehicles by Model Checking, *OCEANS 2009-Europe, OCEANS'09*, Bremen, Germany, 2009, 1-10.
[18] A.R. Willms and S.X. Yang, Real-time Robot Path Planning via a Distance-Propagating Dynamic System With Obstacle Clearance, *IEEE Transactions on Systems, Man and Cybernetics*, 38(3), 2008, 884-893.
[19] Y. Wang, A Hierarchical Abstraction Model for Software Engineering, *International Conference on Software Engineering, In. Proc. of the 2nd int. workshop on The role of abstraction in software engineering, ROA08*, May 11, Leipzig, Germany, 2008, 43-48.
[20] K. Kaljurand and N.E. Fuchs, Verbalizing OWL in Attempto Controlled English, *Proc. of 3rd Int. Workshop on OWL: Experiences and Directions, OWLED'07*, Innsbruck, Austria, 2007.
[21] J.F. Sowa, Common Logic Controlled English (CLCE), *http://wikipedia.org*, 2008.
[22] R. Schwitter, Processable English, Macquarie University, Department of Computing, NSW 2109, Australia, *http://wikipedia.org*, 2008.