Position paper on the infrastructure supporting the development of PDDL

Thomas McCallum *
Centre for Intelligent Systems and their Applications
School of Informatics, The University of Edinburgh
Appleton Tower, Crichton Street, Edinburgh EH8 9LE, UK
t.e.r.mccallum@sms.ed.ac.uk

Abstract

For 5 years now PDDL has been growing in complexity and size. With the languages success the field of Planning and Scheduling (P&S) has grown in the light of the International Planning Competition (IPC), in both complexity and popularity. It is proposed that now is the time to allow the community, as a whole, to drive the progress of PDDL and to create a community outside that of the competition. With this in mind some prerequisites are outlined that are deemed necessary to achieve success in this aim, with both some minor extensions to the language but also with greater development infrastructure.

Introduction

This paper sets out some answers to the issues raised as discussion topics for the ICAPS-03 workshop. It focuses on the PDDL infrastructure and the need to secure greater links with a wider community for the development of the language. There are a number of areas where Planning and Scheduling could greatly increase its desirability to the industrial community starting with some improvements in the way languages and resources are managed.

In the first part of the paper the various questions pointed to by the initial call for papers are discussed. Following this a number of infrastructure changes are proposed, ending up in some suggestions for small extensions to the language to enable more flexibility and knowledge sharing.

Infrastructure

What is the ambition for PDDL?

Although PDDL was initially conceived for the IPC in 1998 by Drew McDermott, it has since become the main language for the Planning and Scheduling community as a whole. As such it is no longer just a 'toy' for use with the competition and is becoming a popular research, and in the near future, industrial standard. These ambitions are necessary to ensure that a sensible, community driven language, that addresses all the issues that come with such a standard is created. Anything less than this and the continuing multitude of small specialist languages will continue.

*I would like to thank the reviewers and John Levine for their helpful comments.

There are many roles for languages in planning whether it be to capture heuristics, to communicate knowledge about a particular domain or to help knowledge acquisition, and as such it is for a standards committee to set the main objectives of the language, guided by what the community are using the language for, rather than one person's ideals, as it is this that will allow the language to continue to become more popular and useful in applied planning. The main feature of PDDL that is important to all these languages is the format and tools for dealing with it. By having the ambition to merge the various syntax and increasing PDDL's flexibility it will be possible to cut the learning and development curve of these language aspects down to a minimum, therefore aiding PDDL's overall appeal. This is in contrast to the belief that there is a need for individual languages for specific planning purposes, which hinders development of complimentary technologies.

There are those that would have PDDL as a language only available to experts rather than general users, but as in other modelling languages such as HTML and VRML, their popularity has been greatly aided by their broad appeal. Users could program PDDL by hand but by making the language available to the community in terms of libraries a number of utilities such as PDDL editors could be written allowing normal users to investigate their own individual planning problems. There are no gains to be had by keeping the language at a research/expert level only therefore it should be an ambition, as the whole premise of planning is to apply to problems in the real-world, to make the language accessible to 'real-world' users.

Are we ready for such a standard?

Even though there are a number of areas that can be improved upon in the language and also a number of pressures to consider in its later development from both research and industrial backgrounds, it is the position of this paper that the community is both ready and willing to accept a common standard. For years there has been development of various specialist languages that confuse and obscure the more general landscape of the field. This is one of the reasons why it is so hard to sell Planning and Scheduling as a serious industrial tool. If a common language was adopted with the ability for the community to make specialist changes for their own particular product it would enable and create a

more uniform and pervasive image to emerge.

Should there be a formal standard committee?

As in other languages such as HTML, C and Java there needs to be a lead direction for a language to go in. It is therefore both right and necessary that a central committee be formed to maintain the coding and theoretical standards of the language. It should be comprised of all the areas of influence, such as research, industrial and commercial. In this way the language will be able to accommodate and present itself to all interested parties. But having made the above points it should also be open to the community to add and update the language for there own particular use. With the right infrastructure a similar effect to the Mozilla project (mozilla 2003) could be achieved. This would enable specialist languages and tools to be written quickly and made available to the community but for them to be made part of the official standard the committee would have to vote on the quality of the change and the real value to the community as a whole.

Should developments be allowed to branch as expressive power increases?

This question rests on the assumption that one community base would be too restrictive to encompass all the needs of a increasing language complexity. With a flexible infrastructure such as the one proposed above there would be little need to branch out. That is not to say though that should a need arise to split the language that it should never be considered. In this early stage though it would be unwise and lead to the continuing complex image of the field. With the minor extensions proposed below it should be possible to keep logs of the additions and changes to a standard for the purpose of specialist areas and to allow the community access to it all. The most useful should then naturally be chosen to become part of the overall standard.

Infrastructure Changes

Due to the global nature of the subject the Internet is the most obvious way to promote and develop PDDL. As such an interface that allows users to submit their particular versions of the PDDL language should be made available with the ability to detail changes that they have made. These archives can then be made publicly accessible and if deemed important, additions to the language can be added to the PDDL standard. This common gateway could be used also to develop new tools and foster new discussion on the direction of the language currently only possible via these proceedings which occur too infrequently to drive the language forward.

An example of where a language has had huge success through a community-based approach is the growth of HTML as the standard markup language for the web. The W3C(W3C 2003) organisation is the main body in charge of setting up the standards used on the web but anyone can upload their own tools and updates to the HTML standard to be scrutinised and used by the community. By having mailing lists and newsgroups for people to air their views the whole community feels included in the overall direction of the language. Another advantage of this community-based approach is that documentation becomes more reliable and examples more prolific. This includes the translation of documents into various different languages, there by in itself enlarging the popularity of PDDL.

Another interesting case study is that of the C language(c 2003) that now has a number of working groups. The interesting point here is how it has adapted itself to new technologies such as embedded systems with ease due to its community-based approach. Although there are numerous libraries and extensions available in the public domain, it is still the responsibility of the ISO and IEC to set the C standard. Thereby giving a strict standard to adhere to but also giving maximum flexibility.

There are disadvantages though with the community-based approach. For instance if there is a disagreement about a particular implementation for the standard then this may cause a split in the community. At the moment as there is only a very small number of people who decide on how PDDL grows this is not a problem. A solution may be that some sort of version split may occur but the differences could be allowed to be switched on/off depending on a requirement flag as happens at the moment. Therefore in a way the community will produce its own solution to the problem in its flexibility. Another disadvantage is that there will be a lot more documentation to be done and laziness may creep in to certain versions. This though has also been overcome by other communities by allowing the less competent programmers to debug and document any new versions of tools and the language that come out. Therefore the scale of the community is what keeps these factors in check.

In order to properly archive the expected mass of versions it would be necessary to create add some meta data to the current language. These extensions, proposed in the next section, would allow a systematic library to be created, detailing standard and complexity of the language in use.

Extensions

With the above infrastructure ideas in mind a number of extensions to the language would seem necessary to be able to maintain integrity of the language. These extensions would also help the ability to share knowledge of domains amongst various Planning and Scheduling software.

The first extension is that of language version. With the formation of a committee one of the first things necessary would be to create a numbering system that reflected the distance of the new language concept from that of the standard. An example might be as follows:

[standard version e.g. year.month.day of release] - [specific version of change]

This particular example allows the programmer reading this to determine how current the language version is in terms of the PDDL standards but also a particular version number for the specialisation which would be maintainable by the author.

The next few extensions area also to do with authoring of the language. An indication of the maximum PDDL level reached would be useful so that programs written for a specific level do not try and read above that level which could

cause problems in an automated environment. Finally on this theme an indication of where the language originated and a contact email would be helpful for extra development purposes. These fields would be optional as in HTML but authors would be wise to use them to help third-party users adoption of their specialist modifications.

Therefore a new meta keyword might look as follows:

(:meta 02.11.14-1.11 3 'University Of Edinburgh' 'author@ed.ac.uk')

Translated as the community standard set on the 14-11-2002 altered by University of Edinburgh, released again at version 1.11 which goes up to PDDL level 3. The person to contact for any questions is author@ed.ac.uk.

Just as in HTML this sort of information will allow programs to inter-operate and programmers will be able to write their software to any degree of specialisation that they desire but still be able to contribute to the community as a whole. This sort of information might also generate greater interest in the generation of a hybrid between PDDL and XML which can handle this sort of meta-information in a predictable manner.

Conclusion

In this paper it has been outlined that changes to the infrastructure plus a few minor extensions would enable the language to present itself in as both a research and industrial standard. It has been argued that this would increase interest from the industrial and commercial communities by simplifying the overall landscape of Planning and Scheduling and allowing greater knowledge sharing between current software. Examples have been suggested from the Internet community where tools and languages have been allowed to grow via a large community making small suggestions and then an overall committee from all areas strengthen the language by using the most popular alterations to set a common standard. This community-based approach does need more planning but it is the way forward if PDDL wants to become a true language standard of the Planning and Scheduling community.

References

2003. Iso/iec c - approved standards. http://anubis.dkuug.dk/JTC1/SC22/WG14/.

2003. The mozilla organisation. http://www.mozilla.org.

2003. World wide web consortium. http://www.w3c.org.