Building Simulation 2009

Eleventh International IBPSA Conference
Glasgow, Scotland
July 27-30, 2009

DOCUMENTATION OF OPEN-SOURCE SIMULATION - ADDRESSING MULTIPLE POINTS OF INTEREST

Dr. Jon W Hand[1],
[1]Energy Systems Research Unit, University of Strathclyde, Glasgow, Scotland

ABSTRACT

As with all large software projects, the support demands of a diverse community of a simulation tool exceeds the means of supply. Interested parties may be users (from novices to experts), support staff (e.g. computing infrastructure technicians, QA specialists), researchers who wish to use or extend a feature of the software, other (possibly remotely located) members of the development team or validation groups who what to ensure equivalence of models.

An open source model for simulation software poses particular challenges. Resources are limited, many developers may never physically meet, and the user community includes novices and those who push at the limits of the virtual physics. Open source supports the discovery of application details yet it does not yet seem to have adopted a business model that is able to amalgamate, preserve and distribute what gets discovered. Although focused on ESP-r, many of the issues raised in this paper are generic and may have a wider applicability.

INTRODUCTION

In June 2002 the Energy Systems Research Unit of the University of Strathclyde in Glasgow announced that the simulation suite ESP-r would become an open source software project under the GNU license. The butterfly that set this storm in motion was the author's reading of "rebel code" by Glyn Moody. This book discussed the benefits and drawbacks of making software available beyond its original development community and freeing others to explore new uses. It argued that one can make a business plan around open source software. The next flap of the wings was passing the book to Prof. Joe Clarke of ESRU who decided to buy into the idea. The ESP-r development community debated this and adopted the (at the time) radical idea that the future of simulation lay in opening it up so that others could build on it and use it for purposes that no one in the existing community could imagine.

The decision to open source carried with it a number of technical and philosophical issues necessitating changes in how the developer and user community worked and communicated. Although many thousands of open source applications exist, few could be described as million line virtual physics laboratories. Many of the challenges confronting the ESRU community since 2002 are unique to the technical domain while others confront open source projects in general.

Decisions were required on how to co-ordinate the contributions of existing and new developers so that the ESP-r distribution maintained its robustness as well as becoming a better platform for exploratory developments in simulation.

The traditional sequence of tasks undertaken by developers and the archivist in ESRU had evolved over a decade. The process might have seemed pedantic to outsiders, but there were few glitches in the million lines of code. The process relied on at degree of paranoia as well as the maintenance of a strict regime within which the actors performed their tasks.

One of the early tasks, when opening up ESP-r, was to scale up without becoming a burden on the archivist:

- Passing code to an archivist in ESRU relied on a manual regime of enforced by convention - these needed to be documented and codified
- Detecting errors in coding and changes in predictions were manual processes. These tests needed to become part of the work flow as well as a design issue for new facilities.
- The transfer of files had a limited audit trail and also required attention to detail.
- User access to the source as a set of compressed archive files on a file server was inefficient. It was Linux and Unix platform-centric.

Many of the above issues were rooted in a person-centred version control system. Clearly what was required was a software based version control. ESP-r, as a community, was a late adopter of version control.

In 2001, 2003 and 2004 source code repositories were implemented at different development sites. These made use of CVS (concurrent versioning system) and were used to co-ordinate group coding and testing cycles. This diverse testing ran in parallel with the archivist's tasks.

In 2005 the repository moved from CVS to a versioning system named Subversion (svn). This

allowed for a clearer audit trail, easier manipulation of files and folders and more options for merging and testing different development branches. It also automated the distribution of information about changes as they happened, provided facilities to view changes made by others and supported free clients on most computing platforms. Word spread that Subversion was an infrastructure that could scale and was robust enough for community development.

What had been learned in the testing phase was being written down and demonstrations given within the community. Make no mistake, there was a lot of scratching of heads. The act of documenting procedures exposes complexity. Why must we jump through all of these hoops? Why can't you just take my five line change?

For this paper, one of the core issues is that documenting procedures requires iteration. Actual practice evolves in subtle ways that are not part of the published checklist. The authors of checklists work from habit rather than reading what they wrote down six months ago. Lesson to be learned? Those who write documentation need others to confirm that the instructions actually work.

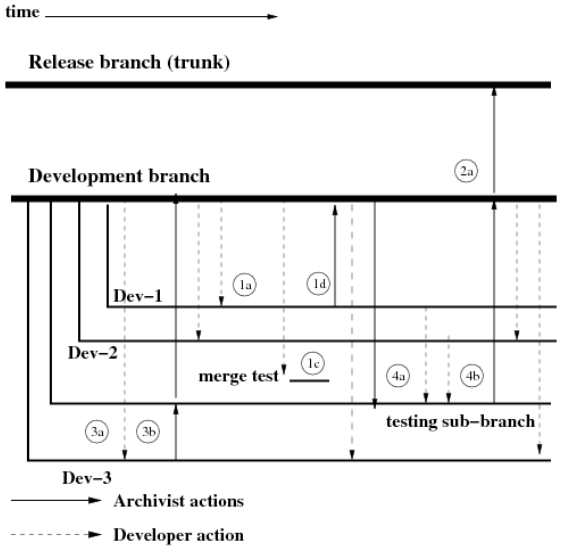

Figure 1 actions sequence

Virtual servers

Once the community was more-or-less comfortable with Subversion, the next step was to move it to an externally hosted environment with a domain owned by the community (esp-r.net) in 2006. From a core of developers who had access to a Sun Solaris box named sigma to scores of developers and hundreds of users relying on an anonymous server is quite a transition.

Another mark of open software is adaptation to the goals of a community rather than its founding authors. This stage of 'letting go' is also a useful proof that procedures are robust and are largely independent of the individuals. Thus, in parallel with the move to a global repository, the archivist role transferred from Joe Clarke in ESRU to Ian Beausoleil-Morrison initially at Natural Resources Canada and later at Carlton University in Ottawa.

The development community has expanded to more than 60 development branches although some of these branches evidence little activity others support joint projects and can be subject to a half-dozen commit/test cycles in a day. The number of commits will have passed four thousand since the SVN repository was established.

ENFORCED TESTING

Although the initial regime was supported by familiarity within a small community, the introduction of new and unknown talent required the core developers to re-consider procedures:

- each commit introduces the risk of errors and there is a considerable benefit in identifying these as early as possible
- developers who are focused on one facet of ESP-r may not realise that their work may have unintended consequences,
- the audit trail built into SVN is a powerful aid to testing new features

The testing method adopted was multi-faceted. The code had to pass a syntax check, it had to compile on multiple platforms, over a hundred example models had to be installed successfully and the predictions from simulations on test models had to be within a specific tolerance.

Each of these facets required that the user community understand the nature of the tests, how the tests were judged, how they should be run and how to contribute new tests as facilities were extended. What was initially a ritual undertaken by the core developers needed to be codified so that others could participate.

In terms of the source code, differences in syntax with the prior state of the code were flagged as seen in Figure 2. Initially developers were expected to undertake the syntax check (and some continue to do this) prior to committing changes, this task was later automated and included after commits to the respository were detected. The archivist has the option of requiring reported warnings and errors to be fixed prior to taking the code into the main development branch.

New entrants to the development process often begin with the view that *code that compiles must be correct*. The archivist has a more specific set of requirements – it must compile, if the code relates to an interface the response to users actions must have been tested, code which reads files should be well tested and code associated with calculations must not introduce unexplained changes.

One step in the process was the automatic pre-processing (e.g. local databases and shading calculation files) of 175 example models. If a change

in ESP-r resulted in one of these models failing to install then the specific facet of the model definition that failed could be identified and investigated.

```
Automated test of ESP-r system
Testing commenced on Wed Feb 11 09:29:08 EST 2009

Test summary:
  - Reference version: branches/prj_dev@3114 (Build options: none)
  - Test version:      branches/prj_dev@3876 (Build options: none)

-------------------------------------------------
TEST                                      RESULT
-------------------------------------------------
Static analyis (Forcheck)                   -
Compilation: X11 build                      -
             GTK build                      -
             X-less build                   -
Regression test                             X
-------------------------------------------------
'-': pass; 'X': fail

  . . .
Binary               # Err.    # Warn.    #Info.
-------------------------------------------------
aco                    ( )       (-)       (-)
b2e                    ( )       (-)       (-)
bps                    (-)       (-)       (+)
 . . .
- binary aco: Passed
 Total errors:     reference 0, test 0
 Total warnings:   reference 3, test 1
 Total info msgs:  reference 298, test 241
 Summary of differences:
 -> Info. #  84 --- no path to this statement [ ref 2, test 0 ]
 -> Info. # 124 --- statement label unreferenced [ ref 5, tet 3 ]
 -> Info. # 313 --- possibly no value assigned [ ref 80, test 79 ]
 -> Info. # 315 --- redefined before referenced [ ref 46, test 7 ]
 -> Info. # 323 --- variable unreferenced [ ref 37, test 30 ]
  - New instance in rwsbem.F - line 1240 IOS [appears in: MKSBEM2]
  . . .
 -> Info. # 665 --- ineq of floating point data[ ref 20, test 23 ]
     - New instance in RE-H2-ctl.F - line 378:
       fPEMFC_ON_p .eq. 3. [appears in: RESH2_CTL]
     - New instance in RE-H2-ctl.F - line 378:
       if ( fPEMFC_ON_p .eq. 2. .or. fPEMFC_ON_p .eq. 3. ) then
            [appears in: RESH2_CTL]
     - New instance in RE-H2-ctl.F - line 430:
       ELSEIF ( SOC_battery (>) SOC_electrolyzer_OFF
 . . .
 -> Info. # 675 --- named constant not used [ ref 93, test 92 ]
     - New instance in compressed_cylinder.F - line 547:
       INO_VENT [appears in: COMP_CYL_CHAR]
```

Figure 2 reported syntax differences

```
TEST CASE office_operations (cellular_offices)
 - Folder: cellular_offices
 - Model:  office_operations.cfg
 - MAX error (W) 22.531W (5.4432%)
 - observed in: all_zones:floors:net_flux:month_1(min)
```

Elements with differences	Units	Relative Diff(%)	Absolute Diff
all_zones:heat_gain:annual (max)	W	-5.4432	22.531
all_zones:heat_gain:month_01 (max)	W	-5.4432	22.531
all_zones:heat_loss:annual (max)	W	6.8421	-16.737
all_zones:heat_loss:month_01 (max)	W	6.8421	-16.737
all_zones:net_flux:annual (max)	W	6.8421	-16.737
all_zones:net_flux:annual (min)	W	-5.4432	-22.531
all_zones:net_flux:month_01 (max)	W	6.8421	-16.737
all_zones:net_flux:month_01 (min)	W	-5.4432	-22.531
zone_01:net_flux:annual (max)	W	-1.9168	12.106
zone_01:net_flux:month_01 (max)	W	-1.9168	12.106
zone_01:net_load:annual (max)	W	-1.9168	12.106
zone_01:net_load:month_01 (max)	W	-1.9168	12.106
zone_02:heat_gain:annual (max)	W	3.1329	-14.045
zone_02:heat_gain:month_01 (max)	W	3.1329	-14.045
zone_02:net_flux:annual (min)	W	-11.753	-14.358
zone_02:net_flux:month_01 (min)	W	-11.753	-14.358
zone_02:heat_gain:annual (max)	W	16.092	-18.024
zone_02:heat_gain:month_01 (max)	W	16.092	-18.024
zone_02:heat_loss:annual (max)	W	24.038	-16.093
zone_02:heat_loss:month_01 (max)	W	24.038	-16.093
zone_02:net_flux:annual (max)	W	24.038	-16.093
zone_02:net_flux:annual (min)	W	16.092	18.024
zone_02:surf_06:heat_flux:ann (max)	W	25.455	-18.024
zone_02:surf_06:heat_flux:ann (min)	W	16.092	18.024
zone_02:surf_06:heat_flux:mon1(max)	W	25.455	-18.024
zone_02:surf_06:heat_flux:mon1(min)	W	16.092	18.024
zone_02:thermal_loads:ann (min)	W	-11.753	-14.358
zone_02:thermal_loads:mon_1(min)	W	-11.753	-14.358

Figure 3 reported performance differences

Other errors are only detected at run-time. One automatic test was composed of a set of 143 simulations to be carried out and various reports generated. Some standard reports were compared for differences while XML output was tested whether it was within tolerance and identified specific entities in the model tested that were associated with differences in performance data. An example of this is shown in Figure 3

BLOGS

The audit trail built into SVN become, in effect, a blog for the community as contributions are committed to the repository. Each commit included a message which identified what had changed, what the impact on users and developers was and how the change was tested. An example of this is shown in Figure 4.

COMMIT LOG MESSAGE

- Reduce differences between Install scripts.

  Include Solaris F90 CC library dependencies (libstdc++ is differently named in Sun Studio).

  This commit to allow testing on different platforms (Solaris, 32 bit, Solaris 64 bit, OSX PPC and Intel etc.)

Testing
- Compile GCC 3.4 X11 with and without -m32 compiler directive on a 32 bit Ubuntu 8.4 and run tester.pl script with no differences reported.

CHANGE-SET SUMMARY

A summary of these changes is available at:

http://node9.cvsdude.com/trac/espr/esp-r/changeset/3736

CHANGE LOG

```
   U  branches/Jon_Hand/src/Install
   U  branches/Jon_Hand/src/Install_32
   U  branches/Jon_Hand/src/Install_RH_studio
```

REVISION HISTORY

Revisions:
- http://node9.cvsdude.com/trac/espr/esp-r/log/?verbose=on

Figure 4 notification of change

Each of the branch owners were notified of changes as they occurred. The notification included links so that the differences in the code could be viewed. Once changes were committed to the main development branch each of the individual branches would be updated by the owner of the branch.

What this 'blog' did not support was posting of "here is what I am planning to do". In a large development community there is a risk that developers who are focused on one facet of ESP-r may not realise that others may be working on a related issue until the 'blog' is updated. This level of communication remains an ad-hoc activity and is a weak point in the community.

The above examples are the product of those in the development community who are already adept at doing-the-dance. For those who are joining the ESP-r community there is a need for guidance. Initially this was done as word-of-mouth and via demonstrations. Gradually this was captured and added to the source code repository. Among the documents were:

- Quality Assurance Procedure
- Developers Quality Assurance Checklist
- ESP-r Coding Guide
- An Overview of Subversion for ESP-r Central Users

Given the core developers geeky history, the typsetting language troff was used to format these documents. This is, of course a classic catch-22. The last thing that new developers want is to learn a command line syntax to generate the documents via the troff processor in order to read how the community works.

The community is currently exploring the use of a WIKI to hold some types of documentation about how the process works. A HTML format would fulfil the need to be an open document format and it might be attractive to community members who have thus far avoided contributing documentation.

Code documentation

While the 'blog' has addressed a number of the documentation and communication issues in our diverse community, there is also the issue of documentation within the repository of code and example models.

Ideally, one would judge code documentation by whether others are able to understand the purpose of subroutines, follow procedural logic and understand looping structures. There is also a need for clarity in data structures such as common blocks and local variables as well as the parameters which are passed into and returned from subroutines and functions.

Clarity is a challenge. Extremes tend not to work e.g. ‘ij’ and ‘loop_for_number_of_boilers_counter’ both have drawbacks. If a common block is used a dozen times in one source file does it need to be fully documented each time?

ESP-r contains much legacy code and some of this requires passion to digest even if compilers can do it without complaint. Where the author of the code is still active they may be able to re-code but some code the loss of the initial flow diagram presents a considerable barrier for reverse engineering.

Because of the diverse backgrounds in the development community there are a number of 'styles' of documentation. Being open source, there is limited scope to enforce coding styles. There are, however, guidelines showing acceptable coding conventions and these tend to be enforced by the archivist for new contributions. Extracts are shown in Figure 5.

Documentation

The source code should be well commented and these comments should precede the code fragments to which they relate. Such comments should be used judiciously and grouped in a manner that illuminates rather than obscures the code. Spacing should also be used judiciously to logically group blocks of code. Comments may be indented when this clarifies the interpretation of the code. Two examples that illustrate acceptable commenting styles follow.

```
      <preceding code fragment>

C Loop through each of the selected zones and scan the Op
C if it exists. If not, insert default crack connection.
      do 38 izt=1,izn
        iz=ivals(izt)
        <following lines of code>

C Increment pointer to the current zone.
        nodeforcurrent=nodeforcurrent+1
```

Local variables should be documented in the subroutines in which they are used.

Two examples that reveal acceptable styles for documenting variables follow.

```
C Maximum infiltration ('finfmax') and ventilation ('fvntr
C rates (m^3/sec) for each zone. Variable 'icompforinf' is
C number associated with unique infiltration flow paths wh
C is the source zone associated with the largest ventilati
      dimension finfmax(mcom), fvntmax(mcom), icompforinf(
      dimension icompforvent(mcom),isrczforvent(mcom)
      integer icompforinf,icompforvent,isrczforvent
      real finfmax,fvntmax
```

```
      real fCyl_Volume       ! Cylinder gas volume (m3)
      real fCyl_solid_mass   ! Mass of cylinder wall (kg)
      real fCyl_solid_Cp     ! Specific heat of cylinder w
      real fCyl_UA_ambient   ! Heat transfer coeff. betwee
                             !   & ambient (W/oC)
```

Figure 5 suggested code styles

Documentation for users

Viewed from the outside, the provision of documentation for users of ESP-r is of variable quality. It takes many forms. Before interfaces offered contextual help, user manuals were the primary point of reference. The advent of contextual help provides an alternative to reference manuals but it also competes for scarce resources to populate the hundreds of dialogues and scores of menus.

And as was the case with developer interactions with Subversion, seasoned users of ESP-r tend not to use the contextual help and they do not often use the manuals. The former is understandable. The latter seems perverse until one realizes that developers tend to document code and forget that there are manuals which should also be updated.

Within ESRU the traditional approach to skills acquisition includes workshops, mentoring and email communications within the user and developer community. Workshops were either two day introductory courses or three day workshops with time included for user projects and further time on advanced issues such as air flow.

In the context of a two day introductory course the goal was familiarity with the interface, understanding

the building blocks of models and experience of planning and creating models of limited complexity.

- a brief introduction to simulation practice
- a brief history of ESP-r
- a tour of models and simulation issues
- review of translating what is observed in rooms into simulation models
- browse existing models to explore the interface
- review databases
- review of climate patterns
- planning a simple model based on client requirements
- step-by-step creation of a model
- QA of model
- planning and run initial simulation
- discovery of performance patterns
- modifying model, re-run assessments and look for differences in predictions
- adding environmental controls
- understanding control actions
- increasing model resolution
- working practices
- QA techniques

This level of exposure should clarify whether ESP-r might be of interest to the participant and to understand the general nature of interactions within the tool. It would allow a participant to use existing models but would be unlikely to give them sufficient skills to create anything but the most rudimentary models.

In the context of a three day course the goals would be extended to include experience with a range of problems, advanced controls, flow networks and their control as well as data extraction techniques. Several hours would be reserved for individual projects. Participants would be expected to be able to plan and create their own models in consultation with staff.

If this is followed up by periodic explorations then many would develop a competency that would allow the deployment of constrained models that could answer a some design questions but would be much less efficient than staff with greater experience.

The assumption was that workshops would be followed up with mentoring with an experienced user as well as occasional email support and advise.

The presentation materials used in these workshops would often be PowerPoint presentations. The terse nature of most slides is compensated for by the commentary provided by the instructor. On their own, the PowerPoint presentations have been observed to be of limited use for non-participants.

The other mode of workshop presentation is via the live use of ESP-r. Participants would observe the current interface and example models and the instructor commentary would complete the story. Unfortunately, no one thought to video the sessions so that remote practitioners could benefit.

Capturing Expertise

Workshops were considered a success if everyone's first model simulated correctly the first time. And this tended to happen, even for workshops in other countries. In contrast, those who attempted to learn ESP-r in isolation faced a number of frustrations. As admitted earlier, documentation was of variable quality. The absence of the instructor commentary was also a primary difference. In the view of the author this is a critical gap.

Almost no vendor of simulation software writes about and focuses on the importance of method and strategy vis-à-vis simulation. Those who can work magic with simulation have invested in and evolved strategies which greatly leverage the power of the tool they are using. The more general the tool (and ESP-r is very general) the more working practices must be constrained by strategy and the design of models is an art to be tempered by careful planning. Workshop commentary tended to be rich in strategy. But workshops do not scale.

The time and attention needed to notice, understand, explore and eventually deploy useful strategies cannot be limited to those experiencing workshops. The author posited that it should be possible to recast the commentary from the workshops as well as the support that mentors provide for a wider audience.

The approach taken was to place this information within *The ESP-r Cookbook*[Hand 2008]. The first version was distributed in 2004 as a fifty page document. Over time it was extended, new figures added and other updated to about 120 pages. An accompanying volume of Exercises was added in 2007.

In early 2008, while on secondment to Samsung Construction in Seoul, the opportunity arose to significantly update and expand the Cookbook. Firstly, the secondment involved extended mentoring of staff with different backgrounds. Secondly, some topics Samsung wished to explore were not covered fully in the current text. It made sense to invest in the Cookbook so as to deliver information beyond the immediate context.

There was also an interest in working procedures. One of the definitive books on working practice, The CIBSE Applications Manual 11 [CIBE 2000] was approaching a decade since publication. Much of this was still valid but it seemed better to consider the topic in the light of more recent observations.

One observation, which seems not to be covered elsewhere, is that the most successful deployments of simulation are in teams. And especially where team members have evolved their interaction skills as well as an attitude to notice opportunities for delivering greater value. So the Cookbook contains chapters of

checklists and defines possible points of interaction and what each team member might bring to and take form these interaction points. This was noticed by the moderator of the BLDG-SIM list and recommended for users of all tools. While a great complement, it does point out an area of commonality with most simulation environments. Decisions related to the abstraction of physical designs to virtual designs, decisions about what boundary conditions to use and strategies for understanding performance predictions are both generic and transferable.

One topic which has been recently included in the Cookbook is a broad ranging discussion of the art and science of creating and using mass flow networks. Learning about flow networks was almost exclusive to workshops and mentoring. And it is also the case that little is written about the design of flow networks used in other simulation environments.

By the start of 2009 the Cookbook had expanded to 270 pages and 167 figures. There are currently only introductions to the topics of computational fluid dynamics, planning assessments and interpretation of performance patterns. Given that the latter was the focus of several seminars at Seoul National University there will be more than enough evidence to support an extensive discussion in a later edition.

English as a second language

In presenting workshops on ESP-r in locations where English is not a first language, it is clear that the learning curve, already steep for a simulation tool such as ESP-r, is even worse. This forms a significant barrier to the deployment of simulation.

Although there are limited resources in an open source community, the number of people who might translate a chapter is greater than the number of developers. It requires a mutual benefit to be perceived. From the perspective of the development community, the geographic distribution of users may result in additional development resources. For Universities native language training materials may allow courses to be mounted and local expertise to be developed.

The first step has been to work with the Department of Architecture at Seoul National University in Korea to translate the Cookbook. This work is being undertaken by a group of Masters and Phd students with further editing by lecturers and then final revisions by a nation-wide committee. The Korean translation should be available in mid-2009.

Currently the Cookbook is also being translated for use in Italy, France and China. This should greatly assist those working independently in these locations. A recent notable addition to ESP-r documentation is an alternative to the Cookbook by a pair of French users. In these instances it seems that the business model of open source may be up to the task of addressing this aspect of user documentation and, perhaps influencing the deployment of other simulation tools.

CONCLUSION

This paper has discussed several aspects of documentation within the open source ESP-r project. It has noted the variable quality of support materials and efforts to address the needs of the ESP-r community. Some of these have been successful and some still require much more work. The business model places considerable constraints on resources and the clear duplication of information in contextual help and written documentation has yet to be addressed.

This paper has also provided background on the transition of a core developer group from a person-centric set of procedures to a virtual development environment with more than 60 source branches and which has managed to maintain and improve the quality of the code base while scaling to thousands of commits.

Lastly the paper has discussed how a largely-oral tradition of skills acquisition and mentoring has been re-cast into the form of an ESP-r Cookbook and this has generated sufficient interest to lead to its translation into several languages.

Although the paper has been focused on one simulation tool it is likely that other developers will recognise many of the challenges and tactics discussed in the paper. Certainly the traditions evolved within the ESP-r community will continue to evolve. We will want to learn from other groups and it might be that IPBSA can play a future role as a conduit for good practice ideas. Perhaps a future IPBSA conference will address this?

ACKNOWLEDGEMENT

This paper would not have been possible without the willing cooperation of scores of workshop participants and those who have taken part in mentoring with the author and the scores of developers who are part of the ESP-r community.

REFERENCES

Crawley, D., Hand, J., Kummert, M. 2007. Contrasing the Capabilities of Building Energy Performance Simulation Programs., Building and Environment, Elsevier Science Ltd.

Hand, J., Bartholomew D., Irving, S. et. Al. 1988. Application Manual 11: Building Energy and Environmental Modelling. Chartered Institution of Building Services Engineers, London.

Hand, J., 2008. The ESp-r Cookbook. University of Strathclyde, Glasgow, Scotland. http://www.esru.strath.ac.uk.

Marcel, C., 2009. Tome 1- esp-r modelisation et simulation des batiments. Viherio SARL, Grenoble.

Svn repository is at https://esp-r.net/espr/esp-r/branches/development_branch

1