PROPOSAL—CS4115 FALL 2007

RICK HANSON

We[1] propose to write a file format description language which we call *Bellows*. When a program, indeed a file format specification, written in Bellows is compiled, it yields a file reader and a file writer. In the obvious way, these two programs are able to read and write files in the format described by the specification. This entails that the Bellows language is declarative in nature—Bellows programs describe the data, as opposed to describing how the data is to be read and written.

The resultant file readers read the native file data[2] and write out this data in XML format, and the file writers read the XML format (i.e. the format output by the corresponding reader) and write the data in the native format. The scope of this project, however, does not include a facility for changing the data values before they are marshalled back into native format. The idea is that there are already many good choices for performing such XML data transformations.

Information about the type of the constituents of the native data will also be able to be described in a Bellows program. In this case, the file reader and writer will perform type validity checks on the items duly described in the type specification section of the program. The Bellows language processor will be designed to be employed in a setting where the conversion of a large collection of such files will be necessary. However, the scope of this project does not include data relationship verification *between* one native file and another.

The idea for developing Bellows is due to this author's struggles with pre- and post-processing data for a large simulation model called EADSIM (Extended Air Defense Simulation). The scope of the language features can be gleened from looking at an example of an EADSIM file format specification taken from the EADSIM reference manual and which we have included as an attached appendix. Here is displayed some features we want to include in the Bellows language, namely the implementation of variables, literals, a loop structure and the ability to nest such, a conditional, a collection of binary comparison operators, scoping rules (indicated by indentation a là Python), and static type information for data.

Neither the official EADSIM software suite nor supporting software written by the principal or any other party (such as the user base) features any process with the ability to do such as is described in this proposal. Indeed, the EADSIM software is written in C and the file formats published in the EADSIM reference manual are hand-coded from human translators who are reading the C code base. A hopeful consequence of this project would be to provide a proof-of-concept for a more declarative and better organized way for the model and all external tools to process EADSIM data, thereby encouraging the principals to adopt such a capability into the official suite.

Date: Wed 26 Sep 2007.

`rkh2110@columbia.edu`.

[1]This is, of course, the "editorial" We.

[2]The native file is flat, i.e. it is ASCII encoded.

A.12 COM QUERY (.comquery)

*Version: 7 | #ComQuery
BeginMonte EndMonte ONE_OUTPUT_FILE
ScenarioTitle
CommStatName |
HeaderFileName |
OutputName |
PropStatName |
for each ComQuery
    BeginTime EndTime IntervalTime
    RptType TimeIntRptType ReportName |
    PlatTXFlag
    if (PlatTXFlag == CSTATS_SYSNAME)
        #TXPlats
        for each TX Platform
            SelTXPlats |
    PlatRXFlag
    if (PlatRXFlag == CSTATS_SYSNAME)
        #RXPlats
        for each RX Platform
            SelRXPlats |
    NetFlag
    if (NetFlag == CSTATS_NETNAME)
        #Nets
        for each Network
            SelNets |
    NOHEADER TAB COMMA SPACE SEMI_COLON NORMAL TIME_WALLCLOCK TIME_SEC TIME_HHMMSS

Variable Name	**Type/Value**
*Version: 7	Character String
#ComQuery	Integer
#Nets	Integer
#RXPlats	Integer
#TXPlats	Integer
BeginMonte	Integer
BeginTime	Integer (Seconds)
COMMA	1 or 0 (Yes or No)
CommStatName	Character String (limit 80)
EndMonte	Integer
EndTime	Integer (Seconds)
HeaderFileName	Charcter String (limit 80)
IntervalTime	Integer (Seconds)
NetFlag	Integer
NOHEADER	1 or 0 (Yes or No)
NORMAL	1 or 0 (Yes or No)

ONE_OUTPUT_FILE	1 or 0 (Yes or No)
OutputName	Charcter String (limit 55)
PlatRXFlag	Integer
PlatTXFlag	Integer
PropStatName	Charcter String (limit 80)
ReportName	Charcter String (limit 25)
RptType	Short Integer (one of the following) 1 -> COMSPEC_SYSTEM 2 -> COMSPEC_NETWORK 3 -> COMTIME_INTERVAL 7 -> COMCONNSTATUS 8 -> COMCONNTIME
ScenarioTitle	Charcter String (limit 25)
SelNets	Charcter String (limit 25)
SelRXPlats	Charcter String (limit 25)
SelTXPlats	Charcter String (limit 25)
SEMI_COLON	1 or 0 (Yes or No)
SPACE	1 or 0 (Yes or No)
TAB	1 or 0 (Yes or No)
TIME_SEC	1 or 0 (Yes or No)
TIME_HHMMSS	1 or 0 (Yes or No)
TIME_WALLCLOCK	1 or 0 (Yes or No)
TimeIntRptType	3 -> AGGREPORT 4 -> PLATREPORT 5 -> NETREPORT 6 -> PLAT2REPORT