SIGPLAN

August 2, 2010

Elizabeth M. Murphy
Secretary, Securities and Exchange Commission
100 F Street, N.E.
Washington, DC 20549–1090.

Dear Ms. Murphy:

We thank you for the opportunity to submit comments to the Securities and Exchange Commission (SEC) on its proposal S7-08-10 on Asset-Backed Securities. We're specifically interested in the part of the proposal involving the requirement of filing a computer program of the contractual cash flow provisions expressed as downloadable source code in Python. We appreciate the work the SEC has put into the proposal, and believe that our comments will help refine the final proposal to more effectively achieve the Commission's desired public policy goals.

The following comments are submitted by two groups that represent computing professionals. The comments were prepared by the U.S. Public Policy Council of the Association for Computing Machinery (USACM), which serves as a focal point for interactions between the Association for Computing Machinery (ACM) and the US government. The comments were also prepared by ACM SIGPLAN, the Association's Special Interest Group on Programming Languages.

In our comments we make the following key recommendations:

1. Other programming languages may be more appropriate than Python.
2. An interpreted language is neither sufficient nor necessary to achieve the goals outlined by the SEC.
3. Security should be enhanced by executing the code in a "virtual sandbox."
4. All inputs and outputs of the waterfall program must be specified precisely.
5. The SEC should consider developing a domain-specific programming language or library for writing waterfall programs.

USACM and SIGPLAN commend the SEC for this proposal, particularly for the suggestion that issuers of Asset-Backed Securities should supplement the narrative description by filing a waterfall program that can be downloaded and executed by investors. Should you have any questions, or would like to speak with computing experts in areas related to this proposal, Should you have any questions or concerns, please contact Cameron Wilson at our Public Policy Office, 202-659-9711.

Sincerely,

Eugene H. Spafford, Ph.D., D.Sc.
Chair
U.S. Public Policy Council of the
Association for Computing Machinery

Philip Wadler, Ph.D.
Chair
Special Interest Group on Programming Languages
Association for Computing Machinery

ABOUT ACM AND USACM

ACM, the Association for Computing Machinery is the world's oldest and largest educational and scientific computing society, uniting computing educators, researchers and professionals to inspire dialogue, share resources and address the field's challenges.

The ACM U.S. Public Policy Council (USACM) serves as the focal point for ACM's interaction with U.S. government organizations, the computing community, and the U.S. public in all matters of U.S. public policy related to information technology. USACM responds to requests for information and technical expertise from U.S. government agencies and departments, seeks to influence relevant U.S. government policies on behalf of the computing community and the public, and provides information to ACM on relevant U.S. government activities.

ABOUT SIGPLAN

SIGPLAN is a Special Interest Group of ACM that focuses on Programming Languages. In particular, SIGPLAN explores the design, implementation, theory, and efficient use of programming languages and associated tools. Its members are programming language users, developers, implementers, theoreticians, researchers and educators.

usacm ACM US Public Policy Council

SIGPLAN

Comments on Securities and Exchange Commission File Number S7-08-10

Inputs and outputs of the waterfall program

The document proposes that some inputs to the waterfall program take the form of an Asset Data File, which is precisely specified (Section 4, page 190). The document also mentions a number of other inputs and outputs to the waterfall program (summarized in the two paragraphs beginning "Under the proposed ...", pages 210–211).

We recommend that all of the inputs and outputs required by the waterfall program be specified to the same level of detail as the Asset Data File, and that procedures for executing the waterfall program be standardized by the SEC. Otherwise, it may be infeasible for investors to determine how to execute the program. Standards for input, output, and execution are particularly important if the waterfall program is to be driven by other programs, for instance when using Monte Carlo methods.

Pilot program

There are a number of specific issues with the proposal, including security, portability, and reproducibility, which we detail below. For these reasons, we strongly recommend that the proposal first be run as a trial, to gain experience with the relevant issues in advance of a full-scale roll-out.

Language adoption process

No one language is perfect for any application. Ongoing experience and developments may change the balance that affects language choice. In particular, it may prove useful to develop domain-specific languages, as described below. We recommend the SEC put into place a process to periodically review the list of languages that are accepted, and to add or remove languages from the list.

Interpreted language

The document proposes that issuers file a waterfall program to be downloaded and executed on an investor's computer. The document proposes the choice of an interpreted language, in order to avoid security problems connected with hosting executable code on the EDGAR server. This requirement is neither sufficient to achieve security nor necessary. It is not sufficient, because interpreted code may still perform malicious actions on an investor's computer, such as deleting files or discovering and transmitting private information. It is not necessary, because security remedies apply equally well to interpreted and compiled languages.

Requiring issuers to provide source code that is to be executed only on the investor's computer is a sensible precaution. Requiring source code makes it possible to inspect the code for problems. This security precaution can be applied to either interpreted or compiled languages.

One way to deal with malicious code is to require it is written in a "safe" subset of the programming language that cannot perform malicious actions, or to execute the code in a "virtual sandbox" that detects and disables unsafe actions at runtime. We recommend that the SEC specify a security policy along these lines. Again, this security precaution can be applied to either interpreted or compiled languages.

Safe execution of code written by one party on a machine owned by a different party was not a strong concern in the design of Python. It was a strong concern in the design of other systems, including Java and the .Net framework (which supports multiple languages, including C#, F#, and Iron Python).

Programs written in one programming language may execute substantially slower than corresponding programs in another programming language. The widely accessible interpreted implementations of Python and Perl execute more slowly than compiled languages such as Java, C#, F#, and Scheme by as much as one or two orders of magnitude. Slower execution may be an issue for techniques that require intensive computation, such as Monte Carlo methods.

Reproducible semantics

One requirement of the waterfall program is that it should behave identically for the issuer and for all investors that execute it.

Programming languages typically are released in a number of versions, each version of which may behave differently. Python has two major versions (Python 2 and Python 3) each with sub-versions (Python 2.7 and Python 3.1 as of July 2010). We recommend that the SEC should not just specify one or more programming languages in which waterfall programs can be written, but also for each programming language which versions of that language are acceptable.

Each version of a language may possess multiple implementations (interpreters or compilers issued by different organizations) which are intended to agree, but may in fact possess differences. Implementations of Python include CPython, Jython, PyPy, Iron Python, and Unladen Swallow. We recommend that whatever language or version is chosen, the SEC should monitor implementations, track differences, and perhaps recommend those it considers more stable.

A program may also refer to a number of libraries, and these can vary in the same way as versions and implementations. In particular, it may be desirable to develop libraries specifically to support waterfall programs. We recommend that the SEC monitor and recommend libraries in the same way as versions and implementations.

We recommend that the SEC require that each waterfall program be accompanied by: an indication of the programming language and specific version, the libraries required, and their respective version numbers.

Numerical computations can behave differently on different computers. The SEC may wish to give consideration to limiting the set of permissible numeric operations to those that behave consistently on different platforms. We recommend that the SEC should specify that the waterfall program be

accompanied by a set of test cases that can be used to verify that the program is running as intended, particularly with regard to numerics.

An important factor in determining whether a given program behaves as intended is the precision with which the programming language has been specified and the extent of test suites for validating that an implementation of the programming language agrees with its specification. This precision can vary greatly; at one extreme a programming language might have no specification, and at the other it might have a formal mathematical specification. Python has a specification written in English, but its specification has not received the same careful attention as those of Java or C#. A few research languages, notably SML and Scheme, have complete and rigorous mathematical specification. Clarity of specification may be particularly important if the waterfall program is to have legal status, and a rigorous mathematical specification may be particularly helpful in such circumstances.

Deterministic execution

We recommend the SEC should require that the waterfall program be deterministic and stateless. That is, for each specified input it should always produce the same output, and its execution should not depend on other inputs (such as network connections or files other than the specified input files), and it should not change or update other data. Requiring the waterfall program to be deterministic and stateless will facilitate reproducibility, distribution and safety of computations, as well as incorporation into other programs such as for Monte Carlo simulations.

Consistency of program and narrative description

We recommend the SEC should explicitly determine the policy toward resolving differences between the narrative description of the waterfall model and the behavior of the program. If the waterfall program and narrative disagree, the policy should specify which takes precedence. The policy may wish to permit filing updates to the waterfall program and narrative to resolve discrepancies.

Program readability

We believe that the waterfall program will better achieve its goal if it is designed to be easy to read. We recommend that the SEC adopt a set of guidelines for improving the readability of programs. These guidelines should address features including standard formatting and the usage of primitives and operators; the use of meaningful, human-readable names; the appropriate use of data structures; and the use of comments. While not every line needs to be commented, comments should at a minimum identify algorithms used, the functionality and use of procedures (including parameters, return values, errors or exceptions, preconditions, and return values), and describe data structures and their use. The use of naming, comments, and program structure should make clear the correspondence between the waterfall program source code and the narrative description of the disclosure.

Domain-specific languages

An alternative to specifying financial products as a computer program written in a general-purpose programming language is to use a “domain-specific language”, i.e. a language specifically

designed to describe financial products. This may be a purpose-designed programming language, or an existing programming language customized by a set of appropriate libraries.

A domain-specific language would support writing waterfall programs that are shorter and easier to read, and should help ensure consistency between the program and the narrative description. In a domain-specific context, it is easier to address security issues and numeric issues, and to provide a formal and unambiguous definition. Domain-specific programs are often amenable to analysis; for instance, they can be designed to offer additional support for techniques such as Monte Carlo methods.

There is already a considerable body of work on domain-specific languages for financial applications, some of which are already in commercial use. ACM's Policy Office would be happy to provide further details, including putting the SEC in touch with practitioners and researchers working in the area.

Suitability of different programming languages

Here are some comments on the tradeoffs between programming languages that might be considered. We consider a small sample of existing programming languages: Java, C#, F#, Python, Perl. All of these languages have open-source implementations available. ACM's Policy Office would be happy to provide further information (on these languages or others) on request.

Efficiency. Some languages are designed to be more efficient, while others are designed to be more expressive at a cost in efficiency. Efficiency of the chosen language may be important for computationally intensive techniques such as Monte Carlo methods. Java, C#, and F# are designed to be executed efficiently, Python and Perl implementations are significantly less efficient.

Typing. Statically typed languages are generally considered to produce more reliable and easier to maintain code, while dynamically typed languages are generally considered to produce more flexible code and to be better suited for prototyping. Java, C#, F# are statically typed; Python and Perl are dynamically typed.

Security. Some programming languages have been designed with security in mind, and some of their implementations include "sandboxes" that can securely execute untrusted code. Java, C#, and F# are such languages; Python and Perl are not.

Specification. Some languages have a thorough specification intended to precisely clarify the meaning of programs, others less so. Java and C# have had significant effort put into producing precise specifications; F#, Python, and Perl less so. SML and Scheme possess full formal mathematical specifications; F# is descended from SML.

Domain-specific languages. Any programming language can serve as a basis for a domain-specific language by augmenting it with suitable libraries. Experience seems to show that higher-order programming languages such as F# provide a particularly good basis for domain-specific languages. There are financial domain-specific languages available in F#.

Responses to specific questions

This section lists specific questions from the document with answers.

Is it appropriate for us to require most ABS issuers to file the waterfall computer program?

Yes

Should we require, as proposed, that the Rule 424(h) filing include the waterfall computer program?

Yes.

Is it appropriate to require issuers to submit the waterfall computer program in a single programming language, such as Python, to give investors the benefit of a standardized process? If so, is Python the best choice or are there other open source programming language alternatives (such as PERL) that would be better suited for these purposes?

See above for comments on Python and other programming languages. Other choices, such as F#, may be more appropriate than Python. Perl is unlikely to be appropriate, in particular because Perl programs are unlikely to be satisfactory in terms of readability.

Should more than one programming language be allowed? If so, which ones and why?

Possibly more than one programming language should be allowed. We recommend that the SEC formulate explicit criteria that must be satisfied for a programming language to be allowed; and a process for adding and removing languages over time from the list of approved languages. We also recommend that the SEC strongly consider supporting the development of domain-specific languages or libraries for this domain. See above.

Should we restrict ourselves to only open source programming languages or allow fully commercial or partly-commercial languages (such as C-Sharp or Java) to be used? If so, what factors should be considered?

It is appropriate to restrict to languages with at least one open source implementation. This does not rule out C# and Java, which both have open source implementations available.

Are there other requirements we should impose on the possible computer programming languages that are used to satisfy this requirement, other than that such languages be open source and interpreted?

We do not agree it is necessary to require the language to be interpreted. See above.

Under our proposal, issuers would be required to file the waterfall computer program in the form of downloadable source code on EDGAR. Prior to filing, the code would not be tested by the Commission. Would downloading the code onto a local computer give rise to any significant risks for investors? If so, please identify those risks and what steps or measures we should take to address the risks, if any.

Yes, it may give rise to risks. Some of these may be addressed by sandboxing. See above.

Are the proposed input and output requirements for the waterfall computer program appropriate? If not, what type of output and tests should be required for the waterfall computer program? Should the outputs of the waterfall computer program be specified in detail by rule, or broadly defined to afford flexibility to ABS issuers?

The proposed input and output requirements are underspecified, and should be specified in detail. See above.

Should we require comments in the code that explain what each line does? Is this necessary given the narrative disclosure of the waterfall in the prospectus? If it is appropriate, are there any specific explanations we should require?

You should require the code be written to encourage readability and to clarify the correspondence to the narrative. Comments are important, but need not be at the granularity of one per line. See above.

Should we provide a transition period prior to the required compliance date that would allow filers to submit only test filings? Please be specific in your response.

We recommend a pilot program prior to a full rollout. See above.