Swift vs OPM Collections

Ben Clifford, benc@hawaga.org.uk

April 18, 2010

This note talks about hierarchical processes in Swift, and compares them to the proposals for multiple and hierarchical accounts in [OPM1.01] and [OC].

1 Notions of hierarchical processes in Swift

Processes that seem straightforwardly representable in Swift fall into two categories: at the highest level, SwiftScript language constructs (procedures, and perhaps `foreach`, `iterate` and `if` statements); below that, the mechanics of Swift's execution (for example, moving files to, from and between various caches, and interactions with job execution).

Swift provenance work so far has concentrated on the former area, treating all of the low-level behaviour as opaque and exposing neither processes nor artifacts. So (I think) this document will deal with the two fairly separately. The section dealing with low-level processes will need to define not only ways of refinement but also the actual processes and artifacts to be used.

2 Use cases of refinement in Swift

A. Distinguishing what happened at the SwiftScript language level from what happened at the mechanics level.

B. Within the language level, consider a program that consists of nested procedure invocations: outer → inner → helpers → apps, where 'outer' is a driver loop iterating an algorithm over many data sets, 'inner' is the main call to that algorithm, which is partly implemented in SwiftScript using the 'helper' procedure, and partly using external application calls.

We might be interested in the use of our algorithm, which corresponds to being interested in the use of the 'inner' procedure, ignoring who called it ('outer') and the mechanics of the inner procedure ('helpers').

For example, inner might perform a rescaling of a file in a particular image format by using helpers to convert to a standard format, using a standard tool to rescale, and then converting back to the proprietary format - but what we want to query is "what was rescaled?" rather than the nitty gritty. As swiftscript evolves to have more application functionality inside a swiftsript rather than in external apps, this becomes more relevant.

C. "why? provenance" - why was a particular low level operation done? The answer to this might be something that looks like a stack trace in a conventional programming language.

3 Refinement in [OPM1.01] and [OC]

[OPM1.01] permits multiple accounts in a graph, where different accounts may describe the same work using different processes and artifacts. THere is a notion of refinement, where one account α^- describes another account α^+ in more detail (a loosely defined term).

This is sufficient for use case A, where two accounts can be made - one describing the SwiftScript level, and one describing the inner workings. (Swift provenance work to date has concentrated on describing the SwiftScript level and omits inner workings entirely).

For use case B, it is possible to ignore the hierarchy entirely and pay attention only to 'inner' procedures. So the [OPM1.01] model is perhaps sufficient here too.

This model does not address use case C very well. Firstly, its not clear how multiple accounts should be generated in order to provide nested procedure information. At present, Swift generates invalid OPM containing contradictory information in a single account.

[OC] provides a notion of hierarchy in accounts. With this, its possible to encapsulate stack-like information. For example, all SwiftScript procedures at a particular stack depth n could be placed into an account S_n, which refines S_{n-1}.

How does this interact with our use cases?

For use case A, low-level swift workings (perhaps in their own heirarchy) would live in a further refinement below all S_n accounts.

For use case B, calls to 'inner' might now appear in different S_n accounts depending on the definition of 'outer'. This means that to query for all 'inner' processes, it would be necessary to query over all S_n accounts, rather than restricting the query to a particular account. That provides some input, perhaps, to the design of an appropriate query language.

For use case C, does it allow us to extract a stack trace for a given process? Given a process p in account S_n, we know the enclosing account S_{n-1} But that isn't what we want, yet. We want to know the specific process p^+ that enclosed the execution of p: a relation $parentProcessOf$.

Is it possible to infer that relation from [OC] or is further annotation necessary?

4 low-level swift mechanics as an OPM graph

This section discusses provenance information that has not been dealt with at all so far - the mechanics of moving data around and executing processes transparent to a SwiftScript program.

Such information would be less useful to someone interested in the SwiftScript level description of a program and more in the actual operations that put a concrete file in a particular location on a disk somewhere. This has not been the focus of Swift provenance work so far, but is a legitimate area.

TODO this section

References

[OC] http://twiki.ipaw.info/pub/OPM/ChangeProposalMultiple

[OPM1.01] the opm v1.01 specification