Balisage: The Markup Conference

The Open Security Controls Assessment Language (OSCAL): schema and metaschema

Wendell Piez
National Institute of Standards and Technologies / Information Technology Laboratory

Abstract

The Information Technology Lab at NIST is developing prototype formats for machine-readable documentation related to systems security. The Open Security Controls Assessment Language (OSCAL) defines lightweight schemas, along with related infrastructure, for tagging system security information to support routine tasks like crosschecking, validating against arbitrary constraints, and producing punchlists. OSCAL is not conceived as “another big XML application” but as a metaschema. This approach allows us to simplify the design and maintenance of schemas and related tooling; support generation of documentation; produce multiple parallel schemas for XML, JSON, and YAML; and construct conversion tools more easily. Documents and tools leverage basic HTML, or even Markdown, for simplicity even though it limits the expressiveness of what can be directly imported. Conversion is simplified by the metaschema approach, even when multiple schemas apply to a single set of information. We hope that these simplifications will lead to more useful documents.

Table of Contents

Challenges of the systems security domain

> ***Note***
>
> Disclaimer: The opinions expressed in this article are the author's own and do not necessarily represent the views of the National Institute of Standards and Technology.

> This view can be extended to all of problem solving – solving a problem simply means representing it so as to make the solution transparent. [Herbert Simon paraphrasing Saul Amarel]

While it may be useful for markup practitioners to be aware of the Open Security Controls Assessment Language (OSCAL) under development at NIST (the National Institute of Standards and Technology) as an initiative directed at a certain set of vexing problems in information exchange within a particular domain – what may be more interesting is our work providing an infrastructure for OSCAL's ongoing development. As so often in systems design, we find the solution to the hard problems at one level, is solving a simpler set of problems at a lower level. In this case, the lower level represents all the foundational work designing and maintaining schemas,

syntax, and constraint sets for a data format. This is a common challenge for information modelers and tag set developers. But it is impossible to understand why we are approaching this work at this level without understanding something about the peculiarities of the systems we need to support. The world of "systems security documentation" is complex and presents more to learn than an individual can master in a few weeks. Indeed the fact that there is so much to know about it, is what makes it a distinct domain.

What sorts of inhabitants populate this world? Systems security entails analysis, planning, implementation, assessment, authorization and monitoring. All of these activities will be reflected, often, in documentation. This documentation is both complex in itself, and also presents a requirement to be related – and traceable in its relation – to other documents. This information is highly granular hypertext in every meaningful sense of the word, albeit produced for the (laser-) printed page as well as for the screen and automation tool. A security plan may need to cite, in detail, a policy or strategy document on which it is based, or a set of them. An assessment plan for a system must aggregate and integrate information from plans already written for its components. The assessment itself must be able to demonstrate the validity of its assertions in view of a configuration. In fact and in principle, in an IT system, many of the cross checks we promise can be automated. These relations can be choreographed (databases can and have been designed around such information flows), and such a choreography may support the work. While the work depends on this choreography, it stands and falls on its own: a successful design for systems security documentation by no means ensures a success in systems security – while a bad design puts at risk the security it is intended to support. At least according to one analysis, the world of systems security and its documentation needs an interchange format very badly. But a badly designed format could be worse than no format at all.

Salient features of the domain

If you are acquainted with names or acronyms such as Special Publication (SP) 800-53, SP 800-171, FedRAMP (Federal Risk and Authorization Management Program), FIPS (Federal Information Processing Standards) or ISO/IEC 27001[1] – or labels such as AC-1 (Access Control Policy and Procedures) or SI-8 (Spam Protection) – you are probably a specialist in IT systems security and policy. The complexity of the information sets you work with, and the complexity of their relations, both explicit and implicit, is not breaking news. Readers who are not IT security specialists may be able to appreciate this complexity by analogy to other domains, with similar requirements for information management at multiple levels of granularity, across organizational boundaries: the general rule is always that things are more complicated and more nuanced than one might first assume. There are rules, and there are exceptions. And there are rules about the exceptions and exceptions to those. At a superficial material level, things are familiar. A word processor is still a word processor, and a spreadsheet application is still a spreadsheet; well structured data is still well structured, and "narrative" or discursive information – natural (written) language composed and arranged into coherent and interlocking, articulated assemblies, with named parts, but also order among the parts – presents similar challenges for modeling as it does in other fields. But there are some things about the terrain or topography of this particular problem space, that must be understood and respected. None of the characteristics described here are unique to systems security.

1. Impermeable system boundaries – mostly

The working assumption of a language designer must be that if communication channels are open, communication will happen. And while it is perhaps fair to think that this is the normal and desired state inside the system, this does not extend beyond the system boundaries. (That's why it's a boundary.) One system should not be able to read what another one is saying in confidence to someone else, or to itself, or "confidence" becomes meaningless.

Yet there are also times when communication must happen across the boundaries as well, because there are important things to be discussed (if only because it helps if the border guards talk to one another). The need for *qualified and mitigated opacity* to the outside, even while we have transparency within, is the first significant design problem, and practical barrier, we face. (So our

hardest problem in data modeling has been to acquire real-world examples.) Even when an organization is willing to share its security planning data, its policies and protocols already – and for good reason – may prevent it from doing so, even with those who might be in a position to help it. So the need for security hinders the exploration of how to achieve better security. This is a difficult barrier to break down, and it should be: radical and universal transparency is not our goal. Instead, we seek to make this barrier more permeable and flexible, a choice and something to be managed rather than a simple hindrance.

XML (Extensible Markup Language), in particular, has roots in the idea of open data. (Nor is JSON (Javascript Object Notation) any stranger to the idea, of course.) And its use and fitness for domains in which there is every advantage to openness, and no advantage to opacity as such, is well established. It might seem a paradox that openness turns out to be what we need also in systems that are designed not to communicate with one another, except when they must do so seamlessly and with great fidelity to “intention”. We focus on those moments of seamless communication. But we must not forget, that the default setting is and should be not to share.

2. Asymmetric power relations

The paradox is confounded because one of the conversations that must take place is the conversation about system security itself. Organizations cooperating with each other – perhaps they are buying software or contracting with people – may have legitimate interest in one another's internals. And even more than in other domains, these conversations are frequently asymmetric: for example, a software vendor seeking authorization to operate (ATO) in a government agency. At least from a business development point of view, a security plan proposed by a software vendor, integrator, or service provider may be the most important piece of documentation they write all year. To the receiving agency, it is only one of dozens.

This kind of imbalance is not unusual. In other industries, such disparities are managed by finding new ways to distribute and for sharing expertise across organizational boundaries. Possibly a service or consulting sector emerges to mediate. For example, a small scientific research publisher who cannot become expert in JATS – which it must produce in order to be able to submit work to PMC (Pubmed Central), as mandated – can find a conversion vendor who already has the expertise. Such a vendor plays an important role as a node in a system of discrete connection. Nonetheless, economies of scale tend to favor the big players over the small, who cannot afford to outsource, with the effect of magnifying the imbalances in power and capabilities noted above. The large have the advantage of having the margins to invest in these activities. The largest have the advantage of resources of their own to bring to bear.

We think that a solution – or, possibly more realistically, a development conducive to a mix of solutions – to our problems in information exchange supporting systems security – public and private – will need to acknowledge this tendency, again considering it to be neither a good thing in itself, nor a bad thing. Asymmetry may in general favor either the large or the small. What is tricky is to bring everyone forward together. This means looking after the interests of organizations of all sizes, operating legitimately – and it means being an honest broker. Trying to participate as both a player, and a referee, leads to trouble. So we seek to define the rules of a game that players of all sizes, playing together, can referee themselves.

3. Impeded network effects

The two aforementioned characteristics combine together to impede the benefits of network effects. Because not all potential partners can be assumed trustworthy, the assumption must be that system boundaries should be impermeable unless there is a reason for communication to happen. (See salient feature #1.) If they are not impermeable because communications are mutually unintelligible, they can be made impermeable by erecting screens and barriers. And since big movers have large influence (see #2), the network itself shows a tendency to become irregular – not all nodes can be connected easily with other nodes – or even to decompose into smaller networks with few connections between them.

This has the effect of hindering the network effects that are sometimes promoted as one of the most compelling benefits to a standard basis for interchange. (See for example Michael Sperberg-

McQueen's reflection on this idea in his 2002 closing keynote to the predecessor conference of this one, cmsmcq-2002.) In theory, when all nodes in a network can communicate with all other nodes, the addition of each node adds to the network the value of all its new connections, which is to say, the number of all nodes so far. Effectively this results in n-squared scaling of value. But if each new node can communicate not with all others, but only with one, the new value to the network is only the value of the single node's connection, while the cost is the same. Between these extremes is a not-fully-connected (or connectable) network.

Then too, the fact that the default position tends to be “closed” not “open” means that the costs and benefits of developing a shared medium or infrastructure are not shared equally or across an entire community of users. An open communications channel is only useful to those who can find a way to be open. Part of the high price we pay for being closed, is the opportunity cost – what we cannot have because we cannot share.

As an early reader has pointed out, this is especially the case in the systems security domain, where networks become fragmented at many levels. There is a communications gap between regimes conformant to SP 800-53, and those conformant to ISO 27001. Likewise there is a gap wider than a hallway between us XML developers and the web developer in the neighboring cubicle.

4. Multipolarity

This also means there is not one center either. A cloud environment is one with a kind of fractal regular-irregularity, in which our customers, partners, sources, suppliers and service providers come in every shape and size imaginable; and this very much depends on who we are. Similarly, across the domain, there is a proliferation of relevant standards and specifications – and here, we should be thinking not only about technical standards enabling information interchange, and not even of policy documents more broadly, but of *the information itself that is to be shared*, which to a significant extent consists of documents describing, referencing and including “standards” and “specifications”. This proliferation and variation – which indeed is not to be solved by better interchange standards of whatever sort, but encouraged by it – reflects both the complexity of the domain, and the complexity of governance within it. Each governing body that promulgates standards or best practices, from formal SDOs (Standards Development Organizations) through to informal initiatives of communities, seeks to answer a different though overlapping set of requirements for a different – though overlapping – set of problems, kinds of data, even data instances. And an organization that seeks to develop and document its systems security, must often conform or comply to more than one of these different regimes. Not having the choice of one or another, if and as they are mandated, organizations must develop hybrid and synthetic, comprehensive policies, and articulate these for its partners and stakeholders. At the same time, systems security is both a reality, if never absolute or static, and a representation, inasmuch as it is typically attested and demonstrated with documentation describing states of affairs both actual and aspirational. Such documentation could be standardized with significant benefits, both in terms of getting better quality with less effort, and in the usefulness of the information we produce, leveraging it in new ways. Yet at the same time we are aware that security will not be improved by the blind adoption of any technology.

Is it a problem to work and function in a world where any one agent has many partners and even, in some senses, many masters? Not necessarily – though it inevitably poses challenges. The question might be not only, how broadly shared are the means, but also, how broadly shared is the initiative? And can we see to it that responsible players at all levels of scale are able to take meaningful initiative. Assuming it is not possible (or desirable at the cost of having it) for the means and resources to be shared equally (see asymmetry above), this does not mean we need to consider either freedom or capacity (for “security”-related activity or any other) as a fixed, finite resource, that we are unable to cultivate usefully at every level of scale. Other imbalances might matter less if both large and small have freedom to act for themselves.

Things we do not control

The scaling model for any development of a non-proprietary standard or shared technology includes adopters who are relied on to make the technology work for themselves. This is only one of many factors that are effectively externalities that we do not control – even while this project is trying to provide advantages of exactly such a (positive) externality to them, in the form of a robust standard, we rely on them in turn to take advantage of it – to make it an actual standard, not merely a proposal for one. Another factor is the availability of alternatives (technologies or approaches) in an open market, whether and how we work with those.

It must be kept in mind how the IT systems security domain is already technologically mature. Even given the evident need for their work, and despite and because of widespread obliviousness, apathy and (understandable) anxiety regarding the topic among those they work with, security professionals know what job they have to do, and what means are at hand for doing it. This is their focus, and indeed they will resist us technologists as long as our approach is to try taking away (even to "improve") those means, for the simple (if paradoxical) reason that those means are not ends in themselves (documenting security is not the same as having it) – while at the same time, the tools we know have come to be fairly reliable and dependable, for all their flaws, while a new approach remains untested. The problem is familiar enough to give rise to clichés: the chicken/egg problem, or replacing the engine while the ship is underway, etc. In offering OSCAL, we do not control the workflow within which information processing and exchange takes place. Most important, we do not control the data acquisition model: where does the data come from and how does it get into the form we need for it. We do not fail to take note how successful open-technology alternatives for text encoding working in vertical domains (including XML vocabularies such as DITA, JATS and TEI[2], always bring with them at least an implicit model – suited to their respective communities, stakeholders and active constituents – for how the data is ideally to be produced in XML (to an acceptable standard) to begin with. We also do not control the favorite formats of developers, who inevitably have their own perspectives and prejudices – these days, frequently, JSON or YAML (YAML Ain't Markup Language).

This means that either we must develop models that are generic enough to support many or most workflows, either in the center or at the edges, or we must be very flexible with models that can be specialized easily to their use cases. Or both.

The limiting factor of available attention

Finally, security professionals already have to know too much. (Certainly of all the features of this domain, this is one that systems security has in common with others.) With the exception of a few remarkable individuals – those who might break any mold – experts in security should not have to be experts in text encoding, for the simple reason that we need them to be masters of something else. Many of them are already experts in and power users of a range of applications, especially word processors, spreadsheets and databases.

Any set of technical standards, if it is really to facilitate advance, must be good enough to be made useful while it is invisible – or rather, while it is rendered intelligible and tractable by sensible interfaces. This adds considerably to the challenge. Interfaces must in some sense themselves be secured, if only by validibility in principle and demonstration. "Validibility" here implies that not only are we valid, but that we have reason for confidence that we can continue to be valid.

In our case, a strongly limiting factor is the apparent reality that there are key constituencies who are simply not able to embrace a technology that demands they adopt XML, when in their mind they already have a better, more viable and attractive alternative. To be sure the alternative is always a different one, even if it is always "obvious" that the solution should be based in JSON, or a Markdown/YAML hybrid, or a favorite database – or Word or Excel. On the XML side, we have arguments for why XDM (the XML Data Model, as specified by W3C, the World Wide Web Consortium), for example, may be more accommodating than JSON – but we do not agree (much) on what the XML should look like. In none of these cases do debates over why "they" (or "we") might be wrong about any of this, appear to have much if any traction. But given actual requirements for data processing, we have had some success with an approach that is agnostic as to notation (XML or JSON), while taking the mapping and modeling problems that appear

(especially as we try to represent documentary data) seriously on their own terms – fully aware of solutions to those problems in neighboring spaces. Limitations in some respects do not need to hinder – they might even accentuate – progress and improvement in other areas. So tools are the key.

Finally, problems that are not solved, can be demonstrated and dramatized: the first step to a solution. Indeed these problems will not be solved by technologies, but by people reaping the rewards of openness in a context of trust and trustworthiness. We need to stay focused on our text encoding technologies as a means to this end, not as ends in themselves.

An abstract generic approach

Emulating success

Our starting point for design is the idea that a lightweight and versatile alternative to a large and complex schema, would be a more abstract and generic tagging syntax, which permits moving much of the complexity of the application out of the schema and into a higher layer. (See Piez 2011) Distinguishing between a simple set of rules to which everyone would conform, and more complex sets of local requirements to be validated locally, would permit the owners and users of information to fit the tagging to the information, while maintaining a base level of uniformity. That base level can be clean and clear enough to offer graceful and sensible fallbacks, while at the same time the special properties and regularities that are also typical of this data, could be represented and addressed, exposed with hooks to smarter applications built on top of the rudimentary ones.

In this design we have been inspired to a significant extent by DITA (the Darwin Information Typing Architecture); by HTML microformats (Hypertext Markup Language); and by applications of JATS including both BITS (Book Interchange Tag Suite) and NISO STS (the Standards Tag Suite developed at the National Information Standards Organization) that have served to demonstrate the special strengths of JATS in its highly generalized, shared vocabulary. Indeed we feel that to a great extent, the success of "big schema" industry standards – where they are successful – can be traced to the fact that across a landscape of even very disparate applications, there is a common *generalized* description, which serves as a basis and foundation for semantic differentiation within the domain as well as between domains. This implies that, despite all the futility of trying to build out systems-across-systems that are immune to local vagaries in tagging (be it “interoperability” or “blind interchange”, see bauman2011), there has nonetheless been considerable success, across varying industries, driven in large part because *tooling* and capability can be transferred even when data cannot, or not perfectly. We wish to emulate these successes and borrow as much tooling – and prior knowledge and lessons learned – as we can.

As both DITA and JATS suggest in their adaptability across their users, a strength of the abstract, generic approach is that it scales well in complexity. As a counter example: the fact that HTML does not readily suggest this is perhaps not because the microformat approach does not work, but because HTML validation is simply not very robust in general due to its catch-all content models. Any capable documentary system has islands of regularity – frequently metadata – within a choppy sea of mixed content. Simple data can be simple, while complex data should not be more complex that it has to be. Most especially, local variation is tolerated, sometimes by a strategic postponement or deferral of the question of normalization or regularization.

At its heart, OSCAL might be considered another variety of Peter Flynn's "Standard Average Document Grammar" Flynn 2017, this time using familiar HTML vernacular, supported by a generic infrastructure. For similar reasons, the bones of our format are in a set of simple structures for managing and navigating arbitrary "hunks" of content, including constituent logical parts of documentary data. And the hands and eyes are in a relatively rich, while lightweight, set of metadata-descriptive elements. In large part due to simple modeling and reuse of generic structures, when we turned to building out stylesheets and processing for our draft formats, as they have evolved, we were able to demonstrate many of the benefits of a mature stack in immediate returns. And we found that the generic approach to modeling had few downsides.

Where we found we had specific and peculiar validation rules to enforce, as we expected, Schematron proved useful to fill the gaps between local requirements, which in its way vindicated our decision not to try and express every constraint we knew about, in the core set of structures.

An example of a control catalog encoded in OSCAL may be seen here: https://github.com/usnistgov/OSCAL/blob/master/content/nist.gov/SP800-53/rev4/xml/NIST_SP-800-53_rev4_catalog.xml. Figure 1 offers a snippet for a single control, with its subcontrols (control enhancements) elided. SP800-53 contains hundreds of controls and subcontrols, of which any particular application needs to use only a different and varying subset.

Figure 1: A security control, in OSCAL

An HTML display of the same data can be seen here: https://nvd.nist.gov/800-53/Rev4/control/SC-7. Note that the HTML on that page, while it can be generated from OSCAL such as this example, is actually produced upstream from the same source data set that the OSCAL is derived from.

For brevity, the text is curtailed significantly; the actual control includes more parts and a number of subcontrols (“control enhancements”) as well.

```
<control xmlns="http://csrc.nist.gov/ns/oscal/1.0"
   class="SP800-53" id="sc-7">
   <title>Boundary Protection</title>
   <param id="sc-7_prm_1">
      <select>
         <choice>physically</choice>
         <choice>logically</choice>
      </select>
   </param>
   <prop name="label">SC-7</prop>
   <link href="#ref015" rel="reference">FIPS Publication 199</link>
   <link href="#ref072" rel="reference">NIST Special Publication 800-41</link>
   <link href="#ref093" rel="reference">NIST Special Publication 800-77</link>
   <part id="sc-7_smt" name="statement">
      <p>The information system:</p>
      <part id="sc-7_smt.a" name="item">
         <prop name="label">a.</prop>
         <p>Monitors and controls communications at the external boundary of the
            system and at key internal boundaries within the system;</p>
      </part>
      <part id="sc-7_smt.b" name="item">
         <prop name="label">b.</prop>
         <p>Implements subnetworks for publicly accessible system components
            that are <insert param-id="sc-7_prm_1"/> separated from internal
            organizational networks; and</p>
      </part>
      <part id="sc-7_smt.c" name="item">
         <prop name="label">c.</prop>
         <p>Connects to external networks or information systems only through
            managed interfaces consisting of boundary protection devices
            arranged in accordance with an organizational security
            architecture.</p>
      </part>
   </part>
   <part id="sc-7_gdn" name="guidance">
      <p>Managed interfaces include, for example […].</p>
      <link rel="related" href="#ac-4">AC-4</link>
      <link rel="related" href="#ac-17">AC-17</link>
      <link rel="related" href="#ca-3">CA-3</link>
      <link rel="related" href="#cm-7">CM-7</link>
      <link rel="related" href="#cp-8">CP-8</link>
      <link rel="related" href="#ir-4">IR-4</link>
      <link rel="related" href="#ra-3">RA-3</link>
      <link rel="related" href="#sc-5">SC-5</link>
      <link rel="related" href="#sc-13">SC-13</link>
   </part>
   […  more parts, subcontrols …]
</control>
```

Limits of the familiar

But we also knew that this was nowhere near a full solution, the main problem here being that we also face an existing semantic architecture, which already stipulates several kinds of documents in several different roles with one another. (See the OSCAL web site for details on this architecture.) And this was not something we were free to engineer however we might please: the kinds of documents we call “catalogs”, “profiles” (or sometimes “overlays” or “baselines”), planning documents, assessments and assessment reports: all of these are already features of the terrain, well established and understood in their different ways in different contexts. (Thus, offering cases of emergent semantics.) Everyone in the space already knows what all these things are; most importantly, their various functional requirements are tied to the core processes of

security assessment, and are broadly felt and articulated. Yet at the same time, their definitions and the formal features that characterize these information sets are only partly codified and generally not exposed in machine-readable format; and this gap is the space where we can be useful. Whether we offer a solution with elements or attributes matters less in the end than whether we serve a functional need.

Complementing these ideas – and reflecting the core assumption of our design, namely that OSCAL does not need to be everything to everyone to be useful, while on the other hand, it needs to be something to someone – is another design decision. Although OSCAL is abstract and generic enough to be freely adaptable and useful to describe just about any kind of semantics in its data, it is not expected to accommodate any arbitrary document: it is not another generalized documentary encoding solution. This is because, in order to focus on our goals in security automation (including both planning and assessment, considered broadly), we are focusing on the "islands of structure" within the sea of documentary information typically presented. As published (in presentation), such islands frequently appear as tables or structured appendixes. Invariably (they are tables or structured appendixes for a reason) these data sets have been designed with special care to their layout, structure and internal consistency, and hence (if only implicitly) their representations of semantics. Indeed such datasets might already be created and produced in a more highly structured way than a word processor can support, in a spreadsheet or database application. Whatever the workflow or lifecycle, OSCAL does not intend to support its entirety end to end. Rather, it is to be deployed to specific effect at specific points.

So we imagine this information is best produced and managed in some other form, then mapped into OSCAL by automated means. Its authoritative canonical representation, from which OSCAL is derived, might well be another form of XML. Where this is the case, OSCAL is spared having to address a certain set of functional requirements; in particular, since it is focused on the information presented in security control descriptions and not other types of documentary content, OSCAL's catalog and profile models do not need to be able to capture full text transcriptions of arbitrary documents.

From schema to metaschema

What is a metaschema

Within the context of this paper, a metaschema is considered to be any abstracted representation of a set of constraints over (or regularities across) an information set, especially a set of documents. As such, it works like a schema and can serve similar purposes; but a metaschema is not a schema insofar as it is not intended to be implemented directly, that is to say treated as a schema without mediation. Typically, a schema may be derived from a metaschema. This having been said, most any schema will also be something of a metaschema describing itself. You start your metaschema when you start writing comments into your schema.

To the extent that you may have processing requirements for this information, perhaps the metaschema takes on a formal character. Maybe the metaschema emerges first in the form of a set of Naming and Design Rules. And possibly you now embed your schema in the metaschema instead of the other way around. Or derive it by transformation. (Or possibly you code nothing by hand and use an application to produce your schema, in which case you are implicitly adopting something of a metaschema from your schema toolkit.) Once the machine has leverage, that is, over some sort of declarative content with rigorous, schema-oriented semantics, there are many possibilities; and for any set of documentary processing requirements, it will be tempting to conceive an application for addressing them. A formalization offers computational traction over the metaschema constructs.

Once this takes the form of an actual implementation, it subsequently seems quite natural to with to reuse, support and possibly distribute the metaschema itself – among other reasons, to permit schema development across a community more broadly.

A metaschema once formalized can be given expression in many ways, offering different capabilities:

- Embedding and coordinating documentation
- Expressing schema functionality as requirements, and leaving out unneeded schema features
- Given those requirements, easier schema production and incremental maintenance: tweak the metaschema, hit the button to get a new schema
- Easier coordination with related schemas, through modular and/or aligned metaschemas or a single metaschema infrastructure for a family of data models
- More than one schema syntax; so it is not unusual for an XML-oriented metaschema to emit different-but-compatible schemas in different syntaxes
- More than one operational environment can be targeted, assuming data models appropriate to each can be outlined that serve as satisfactory mappings from the conceptual model of the metaschema, with respect to naming and classification, ordering, object or data types, etc.
- A wide variety of spinoff tools – whatever the metaschema might be designed to support

Examples of fully formed metaschema technologies, each with different capabilities suited to its application and requirements space and user community:

- The ODD format ("One Document Does it all") of the Text Encoding Initiative. TEI schemas and documentation, in multiple formats, are generated from ODD instances. This, together with the TEI class model on which it depends, constitute layers in a metaschema architecture.
- The NLM/NISO (National Library of Medicine / National Information Standards Organization) family of schemas and "tag suites" (the name hints at a metaschema) – the several JATS models, BITS, and now NISO STS, are produced from a core set of modules that can be assembled dynamically, as well as surgically altered or replaced in modular fashion, according to a predefined architecture, as described in its documentation (jats-docs).[3] Again, schemas and documentation are maintained and produced together through a single framework. Among other benefits this permits the different models to be closely aligned, sharing definitions where appropriate.
- The UBL (Universal Business Language) Naming and Design Rules are codified and published as a component of UBL; see http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/cnprd02/UBL-NDR-v3.0-cnprd02.html. These rules comprise only a piece of the UBL schema production and maintenance infrastructure (see Holman 2018, which its lead engineer explicitly names a "metaschema").
- Finally, DITA presents an interesting case. Both in its modular/layered schema design, and its encouragement of extension by restriction, DITA itself is very metaschema-like. But a full accounting would take into view all the kinds of validation enabled or supported as well as potential processing as applied to DITA data – which might imply that the measure of success of a metaschema is when it enables all kinds of processing *additional* to its core feature set. Given its roots, that DITA itself should be more a metaschema than a schema, is possibly not surprising – especially if this evolution is as inevitable among capable and long-lived markup technologies as I believe it to be.

The OSCAL Metaschema

In its layered architecture, OSCAL is a family of related document types described by a set of related schemas. It directly reflects an analysis of the security problem space offered by the Risk Management Framework (see RMF 2016). This is not arbitrary or accidental: the RMF provides us a breakdown of the problem space across activities starting with the definition of security- or privacy-related requirements (as we have seen, **controls** in security parlance), and continuing through their refinement and configuration; implementation (i.e., actually performing or delivering on requirements); assessment and auditing. The relations among these various documents serves as a starting point for design of a system of related document types. This complexity, however, also indicates where we need a corresponding simplicity, inasmuch as it suggests how we need our schema development – which is happening on more than one layer at once, at least if/as we must revisit lower layers to make it work at higher layers – to be more flexible than usual. Markup professionals know how complex and demanding is the development, documentation and support of even a single tag set. Security professionals assuredly need something more rigorous, regular and assessable than HTML. This implies that any format facing users (even through GUI interfaces or underlying their spreadsheets) be as simple and unencumbered as possible. One way of achieving this is to move the complexity out of the format

and into the relations between several simpler, “plainer” related formats. This implies more than one usage profile and even more than one schema. However, this precise design to say nothing of the particulars, remains to be determined. In short, we know the complexity of the data processing requirements is such that a flexible and adaptable approach to both the documents, and the modeling exercise itself, is a necessity.

In other words, again we face the paradox of needing bottom-up development to address the broad range of needs, but top-down development to address the specific functional requirements of the domain as it is already defined (and defines itself) with respect to such entities as “catalogs”, “controls”, “enhancements” (also called “subcontrols”) etc. Both together will only be possible if we take an agile design approach well supported by tooling. This should enable us to address – and publish solutions for – the top-down problems while remaining receptive to the bottom-up development.

What kind of tooling did we have in mind, to support such an activity? We started designing the OSCAL Metaschema for the usual reasons one wants a metaschema layer in the development framework: to centralize the design of the schema, with its documentation. We quickly found it had many more capabilities.

Features of OSCAL Metaschema

What makes OSCAL Metaschema distinct is its focus on a particular subdomain within the general space of information modeling, namely that space of negotiation between document-oriented tagging formats (today most conveniently expressed as XML), and application-oriented object notations (today, mostly JSON and YAML, though many others are of course possible). Paradoxically, because JSON and YAML are in some important respects less expressive than XML (inasmuch as they impose more restrictions on naming and cardinality of data objects in context), OSCAL Metaschema must be similarly curtailed in its modeling – only data constructs that transpose easily (using a mapping derivable from the Metaschema), can be described.[4] This makes the Metaschema relatively simple, as far as its modeling capabilities go; and as long as it is not also too confining, simple should also mean easy to use.

Specifically, OSCAL Metaschema requires two significant design sacrifices at the outset. First, in order to ease convertibility and interchangeability at a low level, as mentioned, we decided to describe all discursive data (as roughly distinguished from metadata) using an HTML subset, only very slightly extended to support an essential requirement in processing SP800-53 data (namely, parameter insertion points, as illustrated in statement `sc-7_smt.b` in the example given in Figure 1). The practical consequence is that, whatever namespace you give to your metaschema-defined XML, at the branches it has either (a) simple values, or (b) an HTML subset (mirror). This means among other things we can define a Markdown equivalent for our HTML at the level of prose or discursive text. HTML subsets are also well known to database engineers as workarounds for unstructured prose.

Secondly, we severely restricted the content modeling capability of the metaschema, in order to maintain lossless convertibility with a JSON “mirror” of our XML format (as described below). In specifying this capability we took the conscious approach that less is more, therefore we should not seek to support any capability that we did not actually need, in the data we have.

The OSCALizable subset of XML

An OSCAL Metaschema can be written to describe the tagging of any XML document with the following features. Not by accident, these restrictions taken together have two important effects: they narrow the space of validable XML, restricting it to certain arrangements or “types” (really types of clusters) that impose regularities over and above what a straightforward grammar-based approach to tag validation, needs to impose. In turn this limits the kinds of constructions a Metaschema format must represent. Simultaneously, the structures help serve to make the XML transparently “castable” into an information-equivalent JSON form, both by precluding organizations of data that do not translate well and by providing a kind of “conceptual scaffolding” enabling a mapping.

Any XML that follows these rules is in principle capable of description using OSCAL Metaschema. Some of the rules can be “bent” more easily than others. (For example, it is generally possible to accommodate namespace mixing – if one is willing to make more rules.) But the narrowest definition should include:

- Everything is in a single namespace, which we can bind at the top of documents to names with no prefix.
- We have such a thing as “prose”, which captures a general description of all uses of mixed content (text and element siblings) vs element content in our model. Prose is restricted to a smallish subset of HTML homonyms in the local namespace. At the block level these are `p`, `ul`, `ol`, `table`, `pre` and `img`. Within prose, no attributes are captured (as of yet) except for the minimal values on links and images **href** or **src** needed for functionality. Within blocks, inline elements are similarly restricted to a small subset of HTML tags with loose semantics - `strong`, `em` and the like. This subset casts cleanly to a Markdown syntax equivalent (designed for compatibility with Github Markdown, a common vernacular). This notation can be used to represent OSCAL prose in non-XML environments and converted back to XML when wanted.

  In order to address a functional requirement in catalogs, a single exception is made, namely an element to support parameter value insertion (somewhat analogous to DITA `keyref`) into arbitary running prose (again see Figure 1 for an example).

  Additionally we find it useful to distinguish between prose at the block level and prose inline, insofar as we may wish to permit the latter only in certain circumstances.

  At the edges, the prose model can be tweaked, extended or replaced, but it entails both a schema definition (in the target schema language, XSD for us) and a mapping into Markdown; so it is not trivial. Making the prose model and associated (“rich” or marked-up) data values more pluggable is a possible future work item.
- Prose is always clumped within its parent element: all prose elements at the block level (`p`, `ul`, `ol`, `table` and `pre`) are adjacent (sticking together in a row) within their parent elements. Since they are all known in advance to be prose, they can be selected and handled together, retaining both their ordering as a group (relative to siblings) and their ordering with respect to one another.

  In the JSON representation, a run of prose is cast into a Markdown string. This does introduce a dependency on Markdown parsing in the JSON-to-XML pathway. To ease this requirement, our specification for Markdown is kept as small as possible.
- Parent elements outside prose blocks, also appear only in clumps, by name (i.e. elements can recur, but not after element siblings of a different type). This achieves an orderable mapping into a JSON object model.

  To achieve this we observe the rule (expressed here as XPath 3.0 / XQuery), for every element in an element-only (block-level) context, in source data:

  ```
  let $n := name(), $s := preceding-sibling::*[name() eq $n][1]
  return (empty($s) or ($s is preceding-sibling::*[1]))
  ```

  returning a Boolean value in XPath. (In English: The element is the first of its name to appear inside its parent, or it immediately follows another element of the same name. So A, B, B, C, but not A, B, C, B. But note that we do not have to assert or test this rule over OSCAL data – which always follows it – but only over data whose closeness to the OSCAL subset we wish to assess.)

  This ensures that any element outside of prose appears contiguously with others of the same name, which permits implicit grouping and solves the order/cardinality mapping problem with object notations such as JSON. Because contiguous elements of the same name can be grouped without re-ordering, it is possible to cast the data by groups onto array properties in the target object model, which preserves their relations both with respect to their labeling and, again, their internal order.

  Since the JSON model (and other similar object notations) does not respect ordering of data values outside arrays, the requirement to preserve relative ordering among sibling data values is one of the challenges in bidirectional conversion. A key feature here is that when converting data back into JSON from XML, a process can rely on a Metaschema to determine correct ordering.
- Attribute semantics are limited to a few utilitarian types (XML ID/IDREF, URL types, Boolean etc.), as supported in our target schema languages.

Given these restrictions in XML, and given a set of names for implicit groupings of elements (of the same name) into an object model, we can "cast" from XML into an equivalent model, while also generating schemas for either representation.

The tradeoff here is that not all XML data sets, or even most documents valid to nominal schemas, can be brought into OSCAL Metaschema, as they fail to exhibit these enabling regularities. For those that can, however, development is a straight track. As is possible with XML schema languages such as XSD and RNG, it is possible to produce automatically, on the basis of a static analysis of a document or document set, a metaschema to describe it. Given a document that follows these rules, we can derive the metaschema by running a transformation – then (perhaps after hand refinement) put that metaschema through our set of "build" applications to produce tools for handling the XML – including for mapping other tagging schemes into it. To be sure, such a derived metaschema will help to codify only such semantics as can be detected by inference from element relations, such as which elements are always discrete blocks vs which may occur in line (i.e., mixed with text). Without the vexing problem of mapping elements inside prose, however, this turns out to be quite a bit.

As Appendix A suggests, OSCAL Metaschema equivalent for most any XML tag set, up to a point, is thinkable – the example given there being meant to look like JATS. A Metaschema for an XHTML subset (Hypertext Markup Language in XML syntax) is similarly thinkable.

JSON Alignment

As is well known to students of this problem, XML and JSON have quite different affordances. Yet if we were to convince others to invest in creating OSCAL data, we knew that permitting them a JSON expression was an absolute necessity – whereas being able to provide assurance, on the other hand, that they could always have JSON for free – at the cost only of your XML! – would be a big win. Or alternatively, that the XML is no big deal if you have got your JSON under control. A driving factor for our adoption of a metaschema approach was that it gives us a place to unify and consolidate our approach to both formats.

OSCAL manages this in two ways. First, the metaschema architecture provides a scaffolding for declarations such that everything can be defined together, without any direct reference to its syntactic representations in its eventual outputs. These definitions, that is, are abstracted away, and then carefully bound back, by means of classic layering or indirection, to its operational semantics in the target schema technologies. Because our XSD and our JSON Schema are produced from the same source, alignment between them is a function of the logic that creates them.

In doing so, the Metaschema can address, deterministically and at a systematic level, all the problems of variable expressiveness between XML and JSON. Most importantly among these, it isolates the problem of arbitrary mixed content – either at the element level or within arbitrary mixed elements-with-text as is routine in embedded markup technologies – in such a way that the stress on JSON to represent such information is relieved (see notes on OSCAL's model for prose in the next section). Setting aside any capability for generalized semantic description (so we have nothing like HTML `span` or JATS/BITS `named-content`), instead we exploit the opportunities offered by today's tooling by mandating a simple Markdown format as the notation for prose on the JSON side.

Examples of OSCAL metaschema instances can be seen on our site. The place to begin could be the metaschema for the OSCAL catalog format:
https://github.com/usnistgov/OSCAL/blob/master/src/metaschema/oscal_catalog_metaschema.xml

Figure 2: A metaschema-driven conversion into JSON

As converted by tools produced from the Metaschema, XML from Figure 1 is rendered into JSON as follows. Only the beginning of the control is shown; whitespace has been added for legibility in presentation.

```
{
           "id": "sc-7",
           "class": "SP800-53",
           "title": "Boundary Protection",
           "parameters": {
             "id": "sc-7_prm_1",
             "select": {
               "alternatives": [
                 "physically",
                 "logically"
               ]
             }
```

```
        },
        "properties": {
          "name": "label",
          "label": "SC-7"
        },
        "links": [
          {
            "href": "#ref015",
            "rel": "reference",
            "text": "FIPS Publication 199"
          },
          {
            "href": "#ref072",
            "rel": "reference",
            "text": "NIST Special Publication 800-41"
          },
          {
            "href": "#ref093",
            "rel": "reference",
            "text": "NIST Special Publication 800-77"
          }
        ],
        "parts": [
          {
            "id": "sc-7_smt",
            "name": "statement",
            "prose": "The information system:",
            "parts": [
              {
                "id": "sc-7_smt.a",
                "name": "item",
                "properties": {
                  "name": "label",
                  "label": "a."
                },
                "prose": "Monitors and controls communications at the external
                          boundary of the system and at key internal boundaries
                          within the system;"
              },
              {
                "id": "sc-7_smt.b",
                "name": "item",
                "properties": {
                  "name": "label",
                  "label": "b."
                },
                "prose": "Implements subnetworks for publicly accessible system
                          components that are { sc-7_prm_1 } separated from
                          internal organizational networks; and"
              },
              {
                "id": "sc-7_smt.c",
                "name": "item",
                "properties": {
                  "name": "label",
                  "label": "c."
                },
                "prose": "Connects to external networks or information systems
                          only through managed interfaces consisting of boundary
                          protection devices arranged in accordance with an
                          organizational security architecture."
              }
            ]
          }, ...
```

More tools

Schemas and conversion utilities are only the beginning, and linked documentation is only one output. (See an example at https://pages.nist.gov/OSCAL/docs/schemas/oscal-catalog-xml.)
Other tools we have in the workshop or sketched on the board:

- Produce starter CSS (Cascading Style Sheets) for presentation of valid XML for authoring or production
- Produce starter display HTML XSLT (Extensible Stylesheet Language Transformations, version 1.0 or 3.0)
- Produce Schematron for matching and testing specified subsets (whether families, classes, contexts) according to local requirements
- Produce XSLT for structural validation of JSON/YAML (i.e. a functional analog to JSON Schema) or constraints expressed as queries (i.e., a Schematron analogue for JSON).
- Produce a metaschema to fit a suitable data set, enabling mockup- or sample-driven development

  Alternatively, from an unsuitable data set, produce a Metaschema-oriented diagnostic report

- Further improvements to JSON/object representations supported by Metaschema
  These become possible as more constraints are imposed over Metaschema-defined XML - for example as attribute value uniqueness in scope can be validated, their values can be exploited as object labels
- XForms bindings?

These are all exclusive of the tools that will implement the semantics peculiar to metaschema-defined formats, whether OSCAL or any other.

OSCAL, XML and the SAND trap

In 2014 at this conference, Joshua Lubell presented on the idea – embedded among several others – of a class of application for which (he said) the XML stack was particularly well suited: the SAND, for “Small Arcane Non-trivial Dataset”.[5]The data with which OSCAL is most directly concerned – high-level systems security documentation – is not exactly a SAND in Lubell's sense exactly (while one layer of the OSCAL stack, our catalog format, is indeed designed specifically to fit one of his examples, NIST Special Publication 800-53). Whether data at higher levels of our stack – the so-called profile, implementation, assessment and report layers – qualify as SANDs precisely, is possibly not an important debate. But it is fair to say they are arcane and even esoteric in certain senses of those words.

Yet history suggests that defining such formats is not a simple thing, even in domains such as technical or scholarly publishing where the rewards have been significant. This is because it proves to be not only an engineering problem but fundamentally a set of social and organizational problem as well. OSCAL aims to address this aspect of the situation not simply by floating or positioning yet another vertical XML format, but to enable the flexible and responsive development of more formats like itself in future. In other words, it is designed to serve also as a schema maker's machine tool, or schema kit.

Another way of putting it is that in order to deal with SANDs, we need something both more and less than standards. An encoding standard can be the basis, but a gap typically remains between the adoption of a standard, and the definition of a robust exchange model between parties. OSCAL's Metaschema offers a technical foundation for the specification of formats that can serve the competing requirements of aligning with standards on the one hand, while also addressing local functional needs – including, crucially, the need for integration of this information into security automation systems.

XML for SANDs

Such a “common exchange model” might work like a standard in every way except that it is used by a relatively specialized group of people to a specific set of ends. It may well be a local specialization, adaptation or application of a broader standard or have technical foundations there. What makes it different from a standard is that it is designed to the particular requirements of its constituent parties, over and above the requirements addressed by the standard on which it may be based. It is different from an application in being shared among users who do not have to share a technical stack: only an agreement on the protocol or common language. But as such an agreement or protocol, such a model must itself be stabilized, documented and tested just like any set of encoding guidelines shared between parties, whether standard or entirely ad hoc and private.

The capability offered by such documented, external specification is important in security space as a part of the “immune system” provided by responsive mechanisms to security threats and incidents. The OSCAL catalog and profile models, for example, should not only to provide a basis for standards-based interchange; like other broadly adopted encoding technologies, it must also accommodate local needs for defining and validating both data and constraints over data sets. OSCAL more broadly, as a “family of tag sets with rough comparability”, which is to say the capabilities offered by our schema-making metaschema apparatus, similarly aspires to contribute to the state of the art in fitting the technologies to human and organizational needs for rational and transparent communication.

We envision OSCAL expressions (whether as canonically valid instances, or variant formats) for more than one set of specifications within the security domain -- *standards*, guidelines and best practices that organizations are bound to follow or aspire to. Among these, SP800-53 is the seed we are starting with; but our hope is that other similar such documents – which offer similar opportunities and rewards for automation – should be similarly easy to express in OSCAL Metaschema, and that their owners will see the considerable advantages in doing so. Being able to integrate these on any platform together is surely a worthy goal. However OSCAL also seeks to support this work at a more fundamental level. There is an intersecting space between word processors, spreadsheets, databases, forms interfaces, web applications, and security automation systems. In this context, an easy and straightforward way to produce a utilitarian schema and set of tools, including tools for mapping and moving data in and out of applications, might be just what developers need. OSCAL Metaschema as a toolkit might have utility even apart from our application space.

Wherever the defined spaces between vertical domains overlap and contend, and it is not clear which of several available alternative formats, even effectively standardized and externalized, provides the easiest course for ongoing development and evolution, developers face a problem. Or to say the same thing in reverse, there are some domains that because they are hybrid, working across boundaries – possibly they mix financial information with patient information, or systems security planning and procedures with systems configuration data – are necessarily a challenge since they are by definition neither quite fish, nor fowl. A lightweight carrier format for pretty-well structured data could fill a niche there – even or especially for data sets that already have (other) standard representations, when they need to cross organizational boundary lines.

A metaschema gives us a way to make throwaway schemas and ad hoc models, while at the same time building out and documenting the core schema, even while it too is still under development. The trouble and expense of tooling to a new data set is lightened by starter-kit tooling provided automatically from Metaschema, such as display stylesheets, or data flatteners, or import/export scripts for databases or spreadsheets.

One document, many schemas

Schemas that are produced from a single metaschema semantics – especially a simple one like OSCAL's – should also have the capacity of being more easily mapped to one another. (In this, the metaschema emulates some of the functionality of ISO/IEC 10744's architectural forms, albeit in somewhat less generalized terms. See Cover 2001.) The possible and even very local and peculiar semantics of data types in one schema (both what XSD calls “simple” and “complex” types, which includes but is not limited to what markup practitioners call content modeling) are more easily fitted to those of another when both use the same language, reducing the problem in the same way. Then too, the common language exposes any non-trivial differentiation between them. Assuming acceptable tradeoffs can be found for managing such differentiation, if both are made using the same metaschema then migrating information sets across formats from one to another, ought to be easier. Indeed this is a general proposition and might be tested by considering other metaschemas besides ours. Presumably it is similarly easier to map between two data sets both supported by TEI ODD descriptions, two documents valid to different flavors of the NLM/NISO JATS family, or two DITA applications.

In other words, when a document is valid to schema A, and a robust mapping exists from all possible A to schema B, then we get validation to schema B for free.

> ***Note***
>
> Of course the second condition is a big “if”, and having a mapping and a mapping that we trust are indeed two separate things. I make the optimistic assumption here that a good mapping can be produced and validated in good faith by knowledgeable parties. Even if this is more the exception than the rule, this does not mean it never happens – and enabling easier development and validation of the maps, might actually help.

There may be a loss in data quality, depending on the mapping. But the channel itself, the “projection” such that our instance of A, can be transitioned securely (under the supervision of a mapping that is explicit and well understood) into an environment set up for schema B, is guaranteed.

We do not yet know whether this particular feature of the system, specifically what we hope or assume should be its adaptability to change, adaptability and migration, will prove to be as critical as we imagine. If no one finds that exposing the data in these formats is worth the effort, then of course downstream benefits will also not accrue; and nothing happens simply because we think it might. But we believe a system designed from the start to be lightweight and flexible, with a built-in forward migration pathway, may have a chance – and may complement, moreover, other approaches where they make sense.

The Rule of Least Power: designing security in

A simple, descriptive, declarative open and legible documentary format can serve as a positive externality for an industry or community. Even when the communications themselves are private, and *even without a common technical infrastructure* (a dependency on a stack), everyone benefits from a common language that enables the expression, paradoxically, of *as little as can be said*, to a purpose:

> Nowadays we have to appreciate the reasons for picking not the most powerful solution but the least powerful. Expressing constraints, relationships and processing instructions in less powerful languages increases the flexibility with which information can be reused: the less powerful the language, the more you can do with the data stored in that language.
>
> Less powerful languages are usually easier to secure. A bug-free regular expression processor, for example, is by definition free of many security exposures that are inherent in the more general runtime one might use for a language like C++. Because programs in simpler languages are easier to analyze, it's also easier to identify the security problems that they do have.
>
> w3cLeastPower

As is pointed out on a Wikipedia page on the Rule of Least Power (ruleofleastpower), this is a refinement of AC-6 (Rule of Least Privilege) in the SP800-53 control vocabulary.

> ***Note***
> CM-7, Least Functionality, might also have been mentioned.

Of course, proponents of XML-based standard vocabularies will and should see here an argument not only for OSCAL or its Metaschema, but for tailored, descriptive and declarative, application-independent data formats in general. While this is certainly the case, one might also go one step further: secure data sets also require the *minimally adequate*, necessary and sufficient encoding for their purposes.

It may be that as a principle, this is overstated. Engineering decisions cannot practically be reduced to formulas, and there may always be a reason to open a trapdoor to a layer down. Yet the reason we define these layers at all, is that we discover their serviceability: having stipulated this, it makes sense for us to observe our own rules, being as scrupulous as we can.

As a reflection, here is the Wikipedia editors' hierarchy of formats, from least powerful (by implication, least problematic and most secure) to most powerful. A few amendments are offered (in bold).

- **Ink on paper (least powerful) and this does *not* mean PDF**
- The plainly descriptive [formats] (such as the content of most databases, or HTML)
    - **Descriptive and declarative languages specified and documented as standards**
    - **Customized or bespoke declarative syntaxes with their specifications and documentation**
    - **Including everything from CSV/spreadsheet dumps through simple JSON up to custom-built XML vocabularies**

  As long as it is declarative and aims to be clean, simple, economical and intelligible
- Logical languages of limited propositional logic (such as access control lists)
  Declarative languages on the verge of being Turing-complete

- Those that are in fact Turing-complete though one is led not to use them that way (XSLT, SQL)
- Those that are functional and Turing-complete general-purpose programming languages
- Those that are “unashamedly imperative”
- **All-purpose feature-rich applications such as word processors and spreadsheets supporting macros and arbitrary execution**

  (A larger category than you might expect)
- **Unsecured, undocumented applications in the Cloud**

Acknowledgements

The technical lead and architect of OSCAL is David Waltermire. He shares project leadership with Michaela Iorga, whose vision and initiative have also been essential. OSCAL is a team effort and this work builds on the work of others. This paper owes much to the support, insight and constructive criticism of Joshua Lubell and other colleagues at NIST as well as anonymous Balisage reviewers.

Appendix A. An example JATSish Metaschema

Another tag set might exploit a Metaschema to useful purpose. The vocabulary here is JATS, except within prose, where we fall into HTML. (Namespace enthusiasts will however take note everything is in a namespace declared by the metaschema.) On the JSON side, the “prose” structure is explicit, where in the XML, it is implied by the sequence of prose elements, which can appear in only one place (as per Metaschema rule) within the sequence of permitted elements.

> ***Note***
>
> At the time of submitting, we are in the midst of redesigning Metaschema; with apologies, we hope the older syntax will illustrate the concept.

The differences in support for cardinality and ordering in XML and JSON are thus mediated.

The metaschema describes this tagging and emanates or expresses schemas in XSD and JSON schema describing the respective notations. Additionally, instances that are valid to either of these formats can be processed by a conversion tool, also produced programmatically from the metaschema, to make the other. It includes a component for producing XML from a Markdown subset equivalent to the HTML-like markup appearing on the XML side.

Again, the JSON equivalent is offered. Note: in the example, escape characters in the JSON representing LF (line feed) characters have been replaced by literal line feeds, for legibility. In reality the JSON is optimized not for legibility but for relatively robust transmission and data exchange.

```
<?xml version="1.0" encoding="UTF-8"?>
<sec xmlns="urn:example:oscal-jats-emulator"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:example:oscal-jats-emulator
    jats-section-metaschema.xsd">
    <sec-meta>
        <contrib>
            <collab>Joint Task Force, Transformation Initiative</collab>
            <role>author</role>
        </contrib>
        <abstract>
            <p>A few lines excerpted from the beginning of the
Introduction to NIST SP800-53, rev 4, <i>Security and
Privacy Controls for Federal Information Systems and
Organizations</i>.</p>
        </abstract>
    </sec-meta>
    <title>Introduction: The need to protect information and information
systems</title>
    <p>There are several key questions that should be answered by
organizations when addressing the information security
considerations for information systems:</p>
    <ul>
    <li>What security controls are needed to satisfy the security
requirements and to adequately mitigate risk incurred by using
information and information systems in the execution of
organizational missions and business functions?</li>
    <li>Have the security controls been implemented, or is there an
implementation plan in place?</li>
    <li>What is the desired or required level of assurance that the
selected security controls, as implemented, are effective in their
application?</li></ul>
    <p>The answers to these questions are not given in isolation but
rather in the context of an effective <em>risk management process
```

```
{ "sec": {
    "sec-meta": {
      "contributors": [
        { "collab": "Joi
          "role": "autho
        }
      ],
      "abstract": {
        "prose": ["A few
Introduction to NIST SP8
Controls for Federal Inf
      }
    },
    "title": "Introducti
information systems",
    "prose":
      "There are several
organizations when addre
considerations for infor

* What security controls
requirements and to adeq
information and informat
organizational missions
* Have the security cont
implementation plan in p
* What is the desired or
selected security contro
application?

The answers to these que
rather in the context of
for the organization tha
necessary, and monitors
```

```
</em> for the organization that identifies, mitigates as deemed
necessary, and monitors on an ongoing basis, risks arising from its
information and information systems.</p>
  <ref-list>
    <ref>
      <citation>National Institute of Standards and Technology (NIST),
Joint Task Force, Transformation Initiative. <i>Security and
Privacy Controls for Federal Information Systems and Organizations
</i>. April 2013,with updates. DOI <a href=
"http://dx.doi.org/10.6028/NIST.SP.800-53r4"
>http://dx.doi.org/10.6028/NIST.SP.800-53r4</a>.</citation>
    </ref>
  </ref-list>
</sec>
```

```
its information and info
    "ref-list": {
      "references": [
        { "citations": "
(NIST), Joint Task Force
and Privacy Controls for
Organizations*. April 20
[http://dx.doi.org/10.60
6028/NIST.SP.800-53r4)."
        }
      ]
    }
  }
}
```

The tooling to support this conversion was not more expensive than the time to develop a Metaschema to describe the tagging. This Metaschema does not describe an OSCAL schema; for purposes of illustration, tag names and definitions are consistent with JATS (while using their own namespace). Because JATS itself describes XML that falls outside the “OSCALizable subset” (section “Features of OSCAL Metaschema”), an OSCAL Metaschema that accounts for all of JATS is not possible. In particular, inline tagging here is HTML-flavored not JATS. However, with allowances made for inline content, a subset of JATS can be described this way; and such a subset of “convertible JATS” might be useful in some contexts. In addition to supporting tooling, for example, such a Metaschema would help define a bridge from a JATS application such as NISO STS, into OSCAL.

As noted earlier, Metaschema syntax is changing even as I write this. More examples of Metaschemas can be found on our web site.

```
<?xml version="1.0" encoding="UTF-8"?>
<METASCHEMA xmlns="http://csrc.nist.gov/ns/oscal/metaschema/1.0" root="sec">

    <schema-name>JATS emulator schema</schema-name>
    <short-name>jats-emulator</short-name>
    <namespace>urn:example:oscal-jats-emulator</namespace>
    <remarks>
        <p>Provides a subset of the JATS 'sec' element.</p>
    </remarks>

    <define-assembly name="sec" group-as="sections">
        <flag name="id" datatype="ID"/>
        <flag name="sec-type"/>
        <flag name="specific-use"/>
        <formal-name>Section</formal-name><description/>
        <model>
            <assembly named="sec-meta"/>
            <field named="title" required="yes"/>
            <prose/>
            <assemblies named="sec"/>
            <assembly named="ref-list"/>
        </model>
    </define-assembly>

    <define-field name="title" as="mixed">
        <formal-name>Title</formal-name><description/>
    </define-field>

    <define-flag name="id" datatype="ID">
        <formal-name>Identifier</formal-name><description/>
    </define-flag>

    <define-flag name="sec-type" datatype="string">
        <formal-name>Section type</formal-name><description/>
    </define-flag>

    <define-flag name="specific-use" datatype="string">
        <formal-name>Specific use</formal-name><description/>
    </define-flag>

    <define-assembly name="sec-meta">
        <formal-name>Section metadata</formal-name><description/>
        <model>
            <assemblies named="contrib"/>
            <assembly named="abstract"/>
            <assembly named="ref-list"/>
        </model>
    </define-assembly>

    <define-assembly name="contrib" group-as="contributors">
        <formal-name>Contributor</formal-name><description/>
        <model>
            <choice>
                <field named="string-name"/>
                <field named="collab"/>
            </choice>
            <field named="role"/>
        </model>
    </define-assembly>
```

```
    <define-assembly name="abstract">
        <formal-name>Abstract</formal-name><description/>
        <model>
            <prose/>
        </model>
    </define-assembly>

    <define-assembly name="ref-list">
        <formal-name>Reference List</formal-name><description/>
        <model>
            <field named="title"/>
            <assemblies named="ref"/>
        </model>
    </define-assembly>

    <define-assembly name="ref" group-as="references">
        <formal-name>Reference</formal-name><description/>
        <model>
            <field  named="label"/>
            <fields named="citation"/>
        </model>
    </define-assembly>

    <define-field name="string-name">
        <formal-name>Name (string form)</formal-name><description/>
    </define-field>

    <define-field name="collab">
        <formal-name>Collaborative Author Name</formal-name><description/>
    </define-field>

    <define-field name="role">
        <formal-name>Role</formal-name><description/>
    </define-field>

    <define-field name="label">
        <formal-name>Label</formal-name><description/>
    </define-field>

    <define-field name="citation" group-as="citations" as="mixed">
        <formal-name>Citation</formal-name><description/>
    </define-field>

</METASCHEMA>
```

Some references

Amarel, Saul. “On the Mechanization of Creative Processes”. IEEE Spectrum 3 (April 1996): 112-114. https://ieeexplore.ieee.org/document/5216589 (restricted access).

[bauman2011] Bauman, Syd. “Interchange vs. Interoperability.” Presented at Balisage: The Markup Conference 2011, Montréal, Canada, August 2 - 5, 2011. In *Proceedings of Balisage: The Markup Conference 2011*. Balisage Series on Markup Technologies, vol. 7 (2011). https://doi.org/10.4242/BalisageVol7.Bauman01.

[w3cLeastPower] Berners-Lee, Timothy, and Noah Mendelson. *The Rule of Least Power* TAG Finding 23 February 2006. http://www.w3.org/2001/tag/doc/leastPower-2006-02-23

[cover2001] Cover, Robin. “Architectural Forms and SGML/XML Architectures”. http://xml.coverpages.org/archForms.htm

Flynn, Peter. “Your Standard Average Document Grammar: just not your average standard.” Presented at Balisage: The Markup Conference 2017, Washington, DC, August 1 - 4, 2017. In *Proceedings of Balisage: The Markup Conference 2017*. Balisage Series on Markup Technologies, vol. 19 (2017). https://doi.org/10.4242/BalisageVol19.Flynn01.

[securitysales2019] “Global Cybersecurity Market to Eclipse $300B by 2024”. Security Sales and Integration, January 17 2019. https://www.securitysales.com/research/global-cybersecurity-market-2024/

[holstege2018] Holstege, Mary. “Metaphors We Code By: Taking Things A Little Too Seriously”. Presented at Balisage: The Markup Conference 2018, Washington, DC, July 31 - August 3, 2018. In *Proceedings of Balisage: The Markup Conference 2018*. Balisage Series on Markup Technologies, vol. 21 (2018). https://doi.org/10.4242/BalisageVol21.Holstege01.

[Holman2018] Holman, G. Ken. “The Universal Business Language ecosystem and the OASIS TC process.” Presented at Symposium on Markup Vocabulary Ecosystems, Washington, DC, July 30, 2018. In *Proceedings of the Symposium on Markup Vocabulary Ecosystems. Balisage Series on Markup Technologies*, vol. 22 (2018). https://doi.org/10.4242/BalisageVol22..Holman01.

Lubell, Joshua. “XForms User Interfaces for Small Arcane Nontrivial Datasets.” Presented at Balisage: The Markup Conference 2014, Washington, DC, August 5 - 8, 2014. In *Proceedings*

of Balisage: The Markup Conference 2014. Balisage Series on Markup Technologies, vol. 13 (2014). https://doi.org/10.4242/BalisageVol13.Lubell01.

[Lubell2016] Lubell, Joshua. “Integrating Top-down and Bottom-up Cybersecurity Guidance using XML.” Presented at Balisage: The Markup Conference 2016, Washington, DC, August 2 - 5, 2016. In *Proceedings of Balisage: The Markup Conference 2016*. Balisage Series on Markup Technologies, vol. 17 (2016). https://doi.org/10.4242/BalisageVol17.Lubell01.

[jats-docs] “Modifying This Tag Set”. In Journal Publishing Tag Library NISO JATS Version 1.2 (ANSI/NISO Z39.96-2019). National Center for Biotechnology Information (NCBI) National Library of Medicine (NLM). https://jats.nlm.nih.gov/publishing/tag-library/1.2/chapter/implementor.html

OSCAL Web site. The Open Security Controls Assessment Language. https://pages.nist.gov/OSCAL/

[piez2011] Piez, Wendell. “Abstract generic microformats for coverage, comprehensiveness, and adaptability.” Presented at Balisage: The Markup Conference 2011, Montréal, Canada, August 2 - 5, 2011. In *Proceedings of Balisage: The Markup Conference 2011*. Balisage Series on Markup Technologies, vol. 7 (2011). https://doi.org/10.4242/BalisageVol7.Piez01.

Risk Management Framework Overview. Updated 2019. https://csrc.nist.gov/projects/risk-management/risk-management-framework-(rmf)-overview

[cmsmcq-2002] Sperberg-McQueen, C. Michael. “What Matters?” Closing keynote address, Extreme Markup Languages 2002. http://cmsmcq.com/2002/whatmatters.html

[teiODD] TEI Wiki page on ODD (One Document Does it All). https://wiki.tei-c.org/index.php/ODD cited 2019 April 17.

Simon, Herbert. *The Sciences of the Artifical*. 1996: MIT Press. Fourth printing, 2001.

[ruleofleastpower] Wikipedia entry on the The Rule of Least Power cited 2019 April 13.

Holman, G. Ken. “Horses for courses: A perspective on an XML vs. JSON discussion.” August 6, 2017. https://www.xml.com/articles/2017/08/06/xml-vs-json-discussion/

La Fontaine, Robin. “Making a difference by processing JSON as XML.” Presented at Balisage: The Markup Conference 2017, Washington, DC, August 1 - 4, 2017. In Proceedings of Balisage: The Markup Conference 2017. Balisage Series on Markup Technologies, vol. 19 (2017). doi:https://doi.org/10.4242/BalisageVol19.

[Lee2011] Lee, David A. “JXON: an Architecture for Schema and Annotation Driven JSON/XML Bidirectional Transformations.” Presented at Balisage: The Markup Conference 2011, Montréal, Canada, August 2 - 5, 2011. In Proceedings of Balisage: The Markup Conference 2011. Balisage Series on Markup Technologies, vol. 7 (2011). doi:https://doi.org/10.4242/BalisageVol7.

Pemberton, Steven. Treating JSON as a subset of XML. XML Prague 2012. http://www.xmlprague.cz/2012/files/xmlprague-2012-proceedings.pdf

Rennau, Hans-Jürgen. “From XML to UDL: a unified document language, supporting multiple markup languages.” Presented at Balisage: The Markup Conference 2012, Montréal, Canada, August 7 - 10, 2012. In Proceedings of Balisage: The Markup Conference 2012. Balisage Series on Markup Technologies, vol. 8 (2012). doi:10.4242/BalisageVol8.

Robie, Jonathan. “XQuery, XSLT and JSON: Adapting the XML stack for a world of XML, HTML, JSON and JavaScript.” Presented at Balisage: The Markup Conference 2012, Montréal, Canada, August 7 - 10, 2012. In Proceedings of Balisage: The Markup Conference 2012. Balisage Series on Markup Technologies, vol. 8 (2012). doi:10.4242/BalisageVol8.

---

[1] FedRAMP (Federal Risk and Authorization Management Program) is a process mandated and operated by the US Government Office of Management and Budget (OMB) for authorizing and assessing systems security for service and software vendors. FIPS is the Federal Information Processing Standards, a set of documented requirements for assurance of computer system security and interoperability, applied across the United States Federal Government and partners. ISO/IEC 27001:2013, *Information technology Security techniques*, is a specification of the International Standards Organization / International Electrotechnical Commission that provides a standardized, generic set of security requirements for organizations of all sizes.

[2] DITA is The Darwin Information Typing Architecture. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita. JATS is the Journal Article Tag Suite: http://jats.niso.org/1.1/. TEI is the Text Encoding Initiative: https://tei-c.org/.

[3] The JATS documentation offers some cogent rationales for maintaining a metaschema. But it does not tell the entire story – for example, while it describes the modular architecture shared by

this particular family of schemas, it says very little about how the documentation itself is maintained and produced.

[4] It bears mentioning that this work was enabled to a great extent by prior work in this area, much of it presented at this conference, as represented in the bibliography. Especially worth mentioning are David Lee's JXON and G Ken Holman's work with UBL as well as the research of Hans Jurgen Rennau and Jonathan Robie.

[5] It might be that this works as an apt label because it evokes the ideas of sandboxes as well as sand castles – while this paper proposes the best way of dealing with SANDy data is to put it in boxes.