*Balisage: The Markup Conference*

# Systems security assurance as (micro) publishing

## *Declarative markup for systems description and assessment*

**Wendell Piez**

**Abstract**

Markup technologies are very general purpose, as reflects their generality of conception. They become interesting as well as useful as they are applied to accomplish goals in the real world. Since principles of generic declarative markup were first applied to accomplishing publishing-related goals in information management, design and application, 25 or 40 years ago, they have repeatedly demonstrated both their generality – they really do work – and their demand for applicability. Get one thing wrong, or leave it out, and the effort sits on a shelf. Design and deploy it carefully and sensitively, and even an inexpensive initiative can pay dividends for years. These systems become sustainable in the context of the sustainable operations of which they are a part.

Decades of experience have shown us how to use declarative markup to sustain publishing operations. Now we have to deal with similar problems of information description, management, reuse across contexts, referencing, tracing, and authentication, only at even larger scales than before, both in size and complexity. This paper proposes some lessons and insights we can bring from our experience with publishing technologies, and suggests how they might be applicable in the growing domain of systems security assurance.

**Table of Contents**



## Context of the conversation

***Note***

Disclaimer: Certain commercial equipment, instruments, or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose. The opinions, recommendations, findings, and conclusions in this publication do not necessarily reflect the views or policies of NIST or the United States Government.

### *Practical successes of declarative markup*

In the form of XML (Extensible Markup Language), for twenty years we have had systems exploiting the principles of declarative markup on a standards basis, with commodity tools. If you count XML's predecessors including both SGML[1] and applications of other technologies that are or can be “declarative” in their approach (such as LaTeX),[2] this history is much longer. This is no accident inasmuch as the roots of these technologies are in typesetting, among other requirements, which presents problems difficult enough to demand we factor out and “layer” solutions addressing challenges in functionality, configurability and maintainability. The layered solution to the problem of maintaining multiple publishing streams from complex aggregated sources, it turns out, is the layered solution to much else as well.

What has not happened? Whether considered as a standard processing stack based on the W3C (Worldwide

Web Consortium) XPath/XQuery Data Model (XDM xdm2017), or only as a data format, XML has not become the single and sufficient solution to all problems. Perhaps it was never, indeed, meant to be, at least not seriously – let us distinguish advocacy from marketing – yet in 2020, even as the broad domains of digital information stretch far beyond what visionaries of 20 years ago imagined, XML or even "layered", declarative approaches considered more generally, are by no means predominant. This is in part because other solutions have emerged to other problems, which have seemed more exigent or demanding than the long-term problems solved by descriptive markup or XML, and which do not entail its overhead. Those of us who work with it know that XML (or the more important principle it stands for) has not "failed"; yet at the same time, it has not altogether taken the field either. Sometimes its successes have been ambiguous. (War stories could be told.) It is probably closer to say that XML has been remarkably, even spectacularly successful in some ways – while in other ways, the future we imagined has not come to pass, or if it has, it appears in a form quite unlike what we expected. (People edit wikis. They use Markdown! War stories could be told.)

### *What is publishing? challenges of documentation in systems security*

Into this ambiguous context we step with a new set of problems, unprecedented and yet (in many respects) familiar at the same time. Systems security assessment, assurance, authorization. Information exchange around systems security – this is not publishing exactly, in anything like the common sense (of making materials available to a public), yet it entails all the same problems of information gathering, organization, exposition, design and presentation for the consumption of readers and consumers, both sentient (people) and automated (machines).

In particular, systems security assessment or RMF-based security assurance activities[3] are not publishing in the sense that they entail creating productions for a general audience. Few of the documents produced by security professionals in their work are "published" in a normal sense. But the size of the audience, or even whether a document is released or disseminated to a public, are not the only defining features of what is "publishing". Perhaps we could refer to this (formal and informal) circulation of formatted office documents as "micropublishing" or targeted publishing. A PDF or Word document, that is, prepared at considerable trouble and expense by dedicated professionals, and submitted for review, whether it be by potential customers, regulatory authorities, or partner organizations, might never be "published", while it is nevertheless subject to all the same functional requirements in information creation, production, management, and tracking – and most especially, for revision cycles and quality control.

Compared to more normal sorts of publishing, this set of activities might work at a higher order of complexity, over faster – and also more extended! – time frames, with larger and more articulated information sets. Even how this is to be done adequately, much less at its best or in its ideal form, has hardly been defined, and its definition is itself dynamic, a moving target, as we learn more about systems and risk migitation and management. At the same time at a certain level we have no choice: this is work that will and must be done, at some level, the only question being how well. In this respect, "publishing" serves as shorthand for an entire range of activities entailing data collection, composition, analysis, review, formatting for presentation and finally the production of "artifacts" for consumption, be they "reports" or "proposals" or "reviews", "specifications" or "assessments" in whatever form - paper, PDF, Office or word-processor formats, web pages, or any other form. In these forms, these artifacts are disseminated to recipients that are able to make use of them for their own processes..

In a paper delivered to Balisage 2019, some of the special challenges of the (so-called) "system security" application domain – as seen from the point of view of a relative newcomer – were described at some length. Additionally, Joshua Lubell's companion paper to this one frames in much greater detail the questions of security assurance processes as a specifically *documentary* activity, and one in which (moreover) the questions of authority, knowledge, sources of knowledge, trust, accountability, traceability and transparency – all issues that have been implicit in every publishing system ever built (whether digital and networked, or by any analog media) – become especially salient. For purposes of this paper, this background is assumed as context.

Indeed, insofar as systems security may also entail both "marketing" and "customer relations" – even while it entails much else – we can recognize that even when it is not formalized to the extent that we see with the United States Federal Government's Risk Management Framework – as one approach to security among others – it is always based in the *appropriate transmission and communication of information between points*. In

other words, it always comes down to a kind of publishing, albeit, as noted, a kind of targeted micropublishing. The details are always different. But a principled approach to the design of information technologies, which works to address one set of thorny problems in information exchange, should provide similar advantages when dealing with another.

## What have we learned from technology in publishing?

### *Your system is someone else's subsystem*

*Self-similarity across scales* - Information ecosystems or ecologies, as noted at Balisage 2018 (Piez 2018), are fractal in organization; one way we know this is by the way we find similar organizations and patterns of organization appearing at many levels of scale. In particular, with publishing workflows we can see a great range of scales and indeed nested scales. Every system is made of systems, with more or less articulated boundaries within the subsystems; and this is true of subsystems too. Essentially, all systems are hybrid systems, and the way to look at any one of them (whether at a "system" level or that of a component) is to consider its interfaces and externalities (including operational and technical dependencies) and especially the way these affect its capacity for maintenance, adaptation and scaling.

Consider for example the banal case of a scientific or scholarly journal. Its communications are carried forward for the sake of producing articles and making them available to a readership. Each article entails a complex choreography of data exchange, as the text of the article is submitted to the journal by its author, reviewed, revised (or rather: rejected in favor of revision), finally accepted and then "processed" (the means varies) for publication. Each of these steps entails one or more communications between parties to accomplish. These communications include the article itself but also the coordination and meta-commentary around it.

In principle this is measurable. A typical small journal might publish four or six issues per year; each of these issues might contain four or six articles, averaging 20 to 30 articles per year. Each of these articles requires a varying (small) number of peer reviews – which we can also count, yielding another number – say, between 50 and 100 peer reviews, per journal, per year. Circulating these peer reviews – and, more importantly, ensuring that documents are revised accordingly – constitutes much (though, not all) of the work of the journal editor perhaps with the help of staff. Since many or most peer reviews will then be returned to an author, for each peer review, there are at least three or four participants in the workflow (author, editor, peer reviewer, staff). To the extent there is attrition, as not all peer reviews and article revisions are carried through successfully, we could quantify this as well. In any case, even without numbers and figures (elapsed time per peer review, to edit it and return it to the author), it becomes clear how the work associated with such circulation is probably the largest limiting factor preventing a journal (on this model) from scaling up to a larger run. Essentially this means that due to the centrality of the editor in the worflow (if only as "peer review conductor") – there is a limit to the effective size of a journal. Making a journal bigger (in terms of production, not circulation) is much harder than spinning off a new journal.

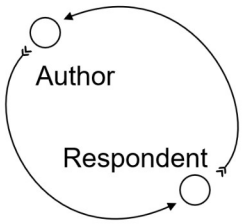
One gating factor here is the relative difficulty of not just peer review, that is, but all the coordination around it. Author, editor, managing editors, peer reviewers: as long as they share no simple platform or standard for the handling of peer reviews, their peer reviewing system is essentially made up of the combination and intersection of all their personal systems, and maintaining communications this way is costly (even while regarded as normal and regular). Migrate peer reviewing to an online system, for example, that consolidates the effort of reviewing and tracking, and the numbers shift. Scaling is now easier – albeit other limiting factors such as availability or motivation, might remain.

Similarly, further along the lifecycle, once journals start being aggregated together, peer review is no longer a limiting factor. Once articles enter "post publication", they are more like black boxes, identifiable by metadata but not requiring intervention for particular cases (as peer review might). So publishers or aggregators of published information can pull together multiple issues of multiple journals, and the scaling bottleneck posed by peer reviews (or again more precisely, by the interventions they require), does not apply. Similarly – but differently – this particular bottleneck does not apply to other sorts of publishing such as trade or monograph publishing, where peer review and the revision process are done and managed differently.

These differences and distinctions between systems and the industries they serve, are driven and defined by their information processing requirements, and the difficulty and expense of those requirements – most of

which are the expenses of time and expert attention. What we do not have, to take account of all this, is a science of workflows,[4] which is to say a set of principles and governing ideas for how workflows actually work: a branch of sociology and economics, but with a technical aspect insofar as workflows lend themselves to quasi-formal definition, once (for example) we start to specify the details of inputs and outputs.

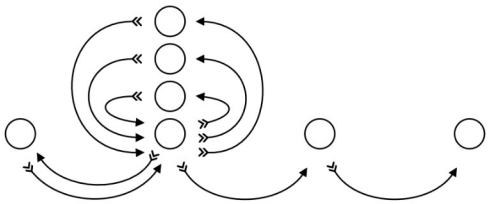
**Figure 1: Elements of workflow**




One principle of a science of worflow should be the concept of exchange. A party gives something to another party, who gives something in return. This response typically triggers another action, perhaps an iteration with a modified input. Such exchanges might be the "elements" or conceptual primitives of a systematic accounting.

When the parties are reciprocal and co-equal, and contents of the exchanges are equally contributed by both, we have a basic "Correspondence" pattern.

**Figure 2: Peer review pattern**



"Peer review" is one pattern in a (potential) pattern language to describe information workflows. Others might be "bundle"; "enhance" (e.g., provide metadata); "test" or "confirm"; "publish" (maybe both "push" and "pull") and so forth.

What a science of workflow would enable us to see would be the articulated joints of these related processes – where there are hand-offs in responsibility, and where there are requirements and capabilities for (what should be called) *transformations*, in that their outputs (results, what is "produced") are modifications, translations and enhancements of their inputs (their sources and raw materials). These can be and typically are very specific operations to very local sets of requirements. Nor is this a bug or a problem, in that these particularities are often the entire point of the exercise. (The reason we send an article to a peer reviewer, is that we would like to see what comes back.)

Without such a science, however, there are still things we can know from experience. One reason it is useful to work through a mundane example such as a peer review exercise in a hypothetic journal, is that it dramatizes the underlying reasons why, for example, rates of technological evolution are so uneven. It remains an open question, for example, in 2020, what format or formats are best used to share draft articles (or more precisely, the raw materials of what is to become an article) in a peer review process. Externalities (such as the ubiquity of certain tools or toolkits) push one way, while functional requirements – or even cultural considerations – in

the system itself may push another. Thus there is no perfect solution – journals accept documents created with proprietary word processors because that is all their authors are prepared for. When an unusual consistuency is able to provide a journal with something better (perhaps a community of scientists, or academic scholars with text encoding skills, has tools they prefer) they will often take that. *This is all because the system of the journal is just a subsystem for each of its authors.*

In a fractal landscape, scales are relative, we should expect the same problems or versions of them to turn up in more than one place, albeit differently. While by definition and design, technologies of automation can support scaling, the fact that they also need to be *fitted* so closely – that two cases (say, two journals, or two research articles in a journal) are so similar without being alike – frustrates and even prohibits a cookie cutter approach.

What is true in one domain of information processing, might well be true in others. In publishing, especially technical publishing (however defined), things are always the same except where they are not. Precisely to deal with this variability, experience shows, only a well-defined, openly-specified, and non-proprietary technology can serve as the basis for (open-ended) "solutions", which can be adapted to serve a heterogeneous and changing set of organizations, with their interlocking goals. Experience with academic, scientific and technical publishing indicates that such a technology will be declarative in form. For machine-readable data to be useful "for the duration" (that is the lifecycle of the information, not the system it currently sits in or the form it currently takes), it must systematically and consistently address and characterize the data itself as both artifact and "mechanism". What that mechanism is or should be, is relative to the uses to which we put this information and our needs for handling and processing it.

### *Declarative markup adds value*

As noted, it is possible for a scholar, researcher or "information professional" in 2020 to be entirely on a digital platform, where all the works apart from some odd print artifacts, consumed as well as produced, take digital form, and indeed in which XML, standards, declarative markup and open systems (perhaps broadly including wikis, git repositories, and assorted other forms of hybrid hypertext media on line) are central to the system, while word processors and other page-oriented tools are secondary.

But those are outliers, and almost everyone involved in "authoring" or "research" phases of publishing today still uses word processors, spreadsheets and document formats along with email – almost necessarily a one-off format with little potential for data reuse – for passing their information across systems. "Office documents" as we broadly call them, are the assumed basis for data interchange. Although they exist, journals or publishers that readily work with other kinds of data inputs, even nominally standard formats, are indeed quite rare, and the cases that do so are illustrative. Similarly, despite all the demonstrated advantages of "single source" publishing, publishers that produce anything but pages first (albeit in digitally encoded form, which is to say digital artifacts in PDF or Postscript®), are also quite rare; any web version or archive version is treated as a secondary production. The possible efficiencies to say nothing of the more outlandish potentials of an XML-first workflow, are simply not perceived to be worth it. And indeed they may not be, if the trouble, expense and disruption are certain while the gains are hypothetical.

On the other hand, to claim that markup technologies have had no impact would be to entirely misconstrue what has happened in 25-some years. While to some it may appear that XML's day has come and gone, it keeps coming back and proving its usefulness: indeed it might be said that the larger-scale activities now happening routinely in the publishing space enabling both access, and long-term stability, for collections of unprecedented size and complexity – all of these would be for practical purposes impossible without strategies of declarative (descriptive) markup and the principle of open lingua franca that can be established on its base. Even if markup technologies have not made their way explicitly into the work practices of writers, researchers and editors, it still cannot be said – most especially in the case of HTML (Hypertext Markup Language) and the web, but this is not the only case – that these technologies are not significant. Indeed, the reasons we invest energy in producing these representations (indexable, retrievable, massable, filterable, stylable – in ways their word processor documents are not), is because they prove to be so valuable.

To a great extent, this is because the principles themselves are sound. Of course there is nothing especially "XML" about cleanly layered separation, and an architecture that reflects and responds to the requirements of its users not to "paint" information to appear one way or another, but to expose and maintain the information

*as information* – which interestingly we do by describing it. XML, with its associated technologies, is a means to this end.

### *Data acquisition is hard*

Perhaps we all agree already that rich, clean declarative markup is by far the preferred form not only for archiving but for production. Even if we do (and I am not sure I do), the question remains of where that rich information set comes from: how does the XML get there in the first place? By XML, here, we mean of course not XML itself, but a particular kind of XML, as exemplified by documentary formats with descriptive tag sets. For reasons that will become clearer later, I also mean (paradoxically) an "XML" that is not XML at all (in that, as I will discuss, the JSON Javascript Object Notation] variants of OSCAL formats seeks to share the same advantages offered by the XML variants). (OSCAL, the Open Security Controls Assessment Language, is discussed further below.)

In the real world, the answer to the problem of how do we acquire the data to start is generally, with difficulty. When information sources are created and first transmitted as "office documents" (meaning any of a species of word processor, spreadsheet software, whether proprietary or open-standard), conversion into XML can be done (with care, by a skilled operator) "by hand", or it can be semi-automated. Either way it will require additionally a skilled human operator for supervision, definition of data quality, assessment. Given these challenges, automation is expensive, becoming cost-effective only when rates of production are large enough – and rules are clear enough – to reward economies of scale.

In general, as well, while large-scale data conversions supported by externalized providers is a way of making XML, this approach is only affordable or sustainable for certain kinds of information. Systems security and assessment is actually characterized by a great heterogeneity of information formats, including data sets produced by machines as well as by people. This great heterogeneity, plus requirements for sensitive handling of private or confidential data, together make it difficult to outsource the task of data description to a third party or external provider. Data security questions aside (how do you shop for a conversion vendor to reformat your most sensitive and proprietary strategic information?), the combination of high data complexity and distinctiveness (at the levels of subdomain, markets and enterprise) may make for no "sweet spot" for outsourcing data conversions, across the domain of systems security. Partner organizations may need to be able to do this for themselves and each other, without always relying on external expertise.

Other methods of acquiring XML markup also present opportunities along with their own challenges. For example, we can bring XML tools such as structured editors earlier into the workflow; similarly, we can design systems with user interfaces (wizards, forms interfaces) that abstract the structure and its encoding away from the view. Both of these have the effect of providing support for the human operator (writer or editor), while permitting the information to be "XML native"; and the advantages of such system, where they can be used, can be considerable. *Where they can be used* here is the operative qualifier – since this is by no means everywhere: both development and deployment require a level of engagement with both goals and technical means. Generally speaking it is only the more agile and more technically-minded organizations that have been able to take advantage of such opportunities.

Yet there may also be a narrow and somewhat arduous path forward to structured data that goes *through* office documents, not around them. The key here may be templates, which already have the advantage of being the preferred method for much of the industry for capturing their information – largely because the promises of templates *are*, in many respects, the promises of structured data, while the deployment architecture (documents layered with so-called styles) is the same or similar, albeit in proprietary form. In some cases, it may be possible to design combinations of templates, rule sets, transformations and document validations (dynamic checks and feedback) that together can help with the job of data conversion, from a representation internal to the word processor, into an externalized form. For certain very regular and generalizable species or subspecies of documents, in certain organizations, this approach might serve as a useful accelerant to getting structured information into the mix. Word processor as structured editor.

While such a solution is possible, who is going to build the solution, for whose use? Will it have to be producers of the data themselves (which would demand an extraordinary combination of disparate skills), or can there be a market for such development and innovation? If the model of the third-party data conversion

vendor does not serve for addressing the need here, what does?

Moreover, it can be expected that this question will remain acute until we have both adequate specifications for shared data description (to whatever level of "standard" possible), and working systems that respect and implement these standards. Until then, cumbersome data conversions will remain an impediment. Organizations who can insulate themselves at the boundaries, defining for their partners what the specifications of these formats will look like, will have an advantage.

This brings us to incentive structures.

### *Activities are supported by incentive structures*

> "'When I use a word,' Humpty Dumpty said in rather a scornful tone, 'it means just what I choose it to mean — neither more nor less.'
>
> 'The question is,' said Alice, 'whether you can make words mean so many different things.'
>
> 'The question is,' said Humpty Dumpty, 'which is to be master — that's all.'"

One key to understanding these articulations – and how they implicate practical matters such as the serialization format of a data exchange – is to see how the lines and arrows in a diagram demarcate lines of authority and responsibility in a complex system of exchanges, whose most important considerations are not in the details of any single exchange, but rather in the operational context in which all of them take place together, as an orchestrated set.

By "exchange" here we mean more than simply an exchange of data. Certainly, some of these transitions entail data being copied from one system to another. (As a journal article is sent as an email attachment from its authors to the issue editors.) However, exchanges also happen as data transitions in other ways. An editor who assigns to an assistant, for example, a task such as copy editing, may "move" nothing, except assign access control rights in the system (so the assistant can make changes to the copy). Yet to exchange access control is to exchange much else, namely the custody of the article, entirely or in part. (In our example, the assistant may know that some corrections are in scope while others cannot be executed without conducting another loop outward, with the author.)

As an example of the signification of such an exchange, over and above the communication of the exchange itself: a young scholar publishing an article in a leading journal, assumes and acquires thereby some of the authority and credibility (as it were by proxy) of the journal. The published article becomes a line item in a *curriculum vitae* and eventually a tenure application, and as such might be worth as much as or more than the article itself. Of course, this is of no direct interest to readers of the article, who benefit from the scholarship despite the necessarily mixed motives behind it. Indeed in principle, the combination produces a mutual benefit even if not a perfectly symmetrical one. (The journal, and its readers, get the good scholarship. The author gets a shot at tenure.) In any case, several exchanges occur following on the central one (the article's publication) at several levels: exchanges of authority and reputation, as well as notice of interests and alliance.

This kind of thing matters since it shows how these structures are built on and around incentive structures, which is to say combinations of mandates, structured choices, and agreements to cooperate that condition how these systems are built and maintained. (One such agreement to cooperate takes the form of "I will do this task if you employ me and make it part of my responsibility" – which is at one level a significant commitment. And yet some kinds of tasks, it seems, are not routinely accomplished without it.) At question is always not only in what *form* does a data exchange occur (an email, a Word document, a piece of registered mail with a signature, a spreadsheet – or an XML document, valid to a schema?) but also *by what rule is that form determined*, *who makes that rule*, and *whose interests are served* (immediate or long-term) by that rule and its rule set, both actually and apparently? ("Make it a `docx` file because that's what I know how to use.") In our example, the young scholar may happily take on the work of formatting the bibliography, as the journal submission guidelines demand, because she understands this exercise is both valuable in itself (or she wouldn't be asked to do it, presumably), and, in a sense, the price of admission, the cover charge for the club; a demonstration and proof of her willingness and ability to play by the journal's rules and pitch in to the common effort. (It is not entirely uncommon for scholars to find bibliographies in particular as rites of passage.)

Or alternatively (to work this example further) maybe the journal discovers that it can't get good enough

bibliographies from its authors no matter what it does. So the work of reformatting bibliographies is handed to an in-house assistant. The workflow for handling the article is thus articulated, at the point of the bibliography. Custody shifts, with respect to an isolable (rules-bound, thus also “typical”) chunk of data, namely that part of the article (the bibliography or works cited section) with its links or bindings to the rest of the system (conceived in large terms). The workflow becomes more complex and the bibliography becomes a special focus. Submitted to a more stringent set of rules, by an agent or operator who can specialize in them, the bibliography can now be enhanced in ways otherwise impossible, normalized for integration. So the bibliography of the paper is made by this effort to merge more easily with the larger bibliography of the journal or publisher's holdings, not only this bibliography but all of them. The benefits of having such a mega-bibliography grow exponentially as the number of entries grow, so it is worth building as largely as possible if you're going to the trouble at all. However, this comes at a cost: someone must pay for the expert assistance and the technical stack to support it. Someone must learn how to do it. In the real world, someone has to volunteer to take on this responsibility, or there must be a budget to pay someone to do it.

It is not difficult to find examples of such phenomena, which are indeed at the heart of publishing activities or more largely, of business in general. Exchange happens, we can stipulate, when a data set in some form or representation (a “document”, an “article”, a spreadsheet, some sort of formal submission on a template), shifts from one party to another, for an operation to be performed. Submitted to such a process, there may be a new record created, and/or an original record or document may be altered or amended: in any case there is a before/after relation; in a way of speaking, each discrete step can be considered an operation, function or filter.

For each of these, whether machine aided or entirely motivated and performed “by hand”, there is some investment (cost), and some reward. In return for providing its value to the operation, the function or filtering operation must be paid for. Standards-based automation pays for itself when we can make these costs linear (by factoring out costs of design and development), while the benefits remain exponential, whenever we can operationalize such functions or filter to the point that they can be automated.

### *Quality is defined within context*

Another key is to see how, within these transmissions, the question of *quality* is both construed (defined and determined) and maintained. Again within the context of journal publishing, a (nominally) “high-quality” author submission might well take the form of a word processor or “office”document. (Whether it is Microsoft Word, Google Docs or whatever a journal editor might consider acceptable these days.) In this case, the criteria of quality are not in its formatting – how pretty is the research laid out on the page – but rather in its instrinsic properties of argument, evidence and exposition, relating it as subject matter to other subject matter. (Is it original or novel research in its field? Does it relate to the literature in its field in some other meaningful way?) As such, the entire purpose of a produced artifact such as word processor file is to represent its author's work adequately for the purposes of the journal to “publish” it, a complex process that entails among other things (and again, because criteria of quality are extrinsic to this), that the document will be translated into a new form – for example, a PDF for page display, even eventually ink on paper.

In its new form, the reformatted document has “quality” (or one might more properly say “value”) that the original document does not, and is judged accordingly – now, not only for its argument and evidence (its nominal “content”), but also for its aesthetics and accessibility (for example).

In other words, it is worth looking at the before and after states when considering appropriate criteria of evaluation. Before publication, as submitted, we might like the document to look nice on the page; but it is not the page layout by which we judge it. This matters because part of what we intend to do, indeed, is reformat it so it looks different. In other words, we fully expect that the article or work we accept for publication, will be changed in that process, if not in essence (as FRBR “work”[5]) then in representation. Yet as publishers (to say nothing of the production designer), we expect the work to look “better” or more polished (than the author could make it). The publishing enterprise is designed to support such activities through processes that work not simply by adding value but by doing so within the context of new and more stringent criteria for evaluation, changing the definition of “quality” itself.

Moreover, this shift in what might be called the *evaluation context* for determining quality is entirely the point of the workflow, the “refinement” to which the work is subjected. Significantly, this can happen irrespective of

how the worflow's participants constituent – the various players in the exchanges – are more or less oblivious to it. The young scholar is rewarded by publishing the article in the leading journal. Part of the reward is that the article is now listed in indexes; it pops up in searches. Other researchers are led to the scholar's work through these links. The links were made not by the scholar who wrote the article, but by the staff who provided its metadata and aggregators who followed after to consume the refined works (journal articles, issues and volumes) made from the raw word processor documents collected from scholars in the field. The very existence of these links, and the aggregators who make them, may be unknown to the scholar whose paper is being cited. Yet they serve a purpose, and the scholar benefits from them indirectly without knowing about it.

In a way, this is to note again that the plan or design of the "machine" of a publishing enterprise is already larger and more complex than the various machines, people and processes that are embedded in it, sometimes even larger than the participants appreciate. The special opportunities of automating document workflows, where we can do this – which is to say, the opportunities and promise of information technologies to enable things that go beyond what could have been done with ink and paper or even telegraph and telephone – will have to *accommodate* these larger systemic requirements, not work against them. Or they will simply not be viable.

The good news is that we are now at a point where we know that these systems can work, and indeed work well, when their various parts are adequate to their needs and where, just as importantly, contributors to the effort know and understand something about how the system works, and why it takes the form it does. (Even if they cannot see everything all the way to the edges.) Enough documentation projects have subsisted long enough, at various and very different levels of scale and complexity, that we can be confident of what we know about this.

### *Evolution works by little revolutions*

A technological system exists until the day it stops being used. After that day, its relics may subsist, but the system itself does not. But a system can also be renewed over time from the inside, shedding parts of itself (since a system is made of subsystems) and replacing them, as its users continue to use the system, but modify it while using it. At one level of the organizational hierarchy, a system is brought down or replaced; it comes to an end; it is switched out for another. This same activity, seen from the next level up – from the point of view of the larger system in which this subsystem works – the switch out constitutes renewal, not death. Maybe there was a day when you used Eudora or Pine for your email, and now you do no longer. Has your email system died? Or merely migrated? Depending on how we define the system: both.

Just as your system can be someone else's subsystem, all their systems can be subsystems of yours, to the extent that you rely on them to do things for you, that you do not do for yourself. The boundaries in the system are not determined so much by extrinsic factors such as the software or platform on which it runs (especially when so much runs on the cloud), as they are by agency and scopes of responsibility – who is responsible to do what – and the interfaces and functionalities that support this.

New systems do not successfully replace old systems except (ipso facto) when they answer the needs met by the old system, and this can happen in only two ways: either the new system grows out of the old system, as it were within the context of its interfaces, and therefore replaces it organically. Or the new system is engineered to replace the old system by offering the same capabilities, perhaps along with some other definitive advantages such as scalability or ease of use. Again, this might be a single process, looked at from two directions.

So we might consider the way email has replaced sending paper through the post (mail), for most routine transactions. A publisher that once collected stacks of paper manuscripts, now pulls together file sets culled from email attachments. This development happened organically, but it would not have occurred if email had not been designed and extended to serve some of the basic (or "essential") functions of paper mail, even while it is crucially different in other respects. It is worth recalling one of Marshall McLuhan's adages, that the content of any medium is another medium.[6] It is not quite a drop-in replacement – email promised and offers new capabilities beyond what the paper post ever did – but it is capable of many of the same functions and operations.

Any disparity of perspective here tends to be not a disparity of kind but of scale, that is again, of the level of hierarchy at which the problem is viewed. To return to the journal example, to an editor as email user, for example – `editor@journal.edu` – correspondence with contributors and readers is an ongoing and essential process, which must occur for the journal to subsist, somehow or other. This volume of information (correspondence, manuscripts, edited copy, reviews, in-process transcriptions), as a kind of information matrix, is the medium out of which the medium of the journal itself (through a kind of alchemical distillation) is made. From the point of view of the journal – a fish looking at an ocean – the case of its correspondence is distinctive, unique and special to itself. (The editor does not care for anyone else's journal correspondence, nor are they expected to.) As a team, the journal staff undertakes the responsibility of supporting this exchange with the people they wish to reach. (This is what it means to produce a journal.) Does this mean they need to develop their own postal service or messaging platform? No: in the real world, what they do is necessarily what their correspondents and partners in exchange (authors, readers) already do. In other words they do not and indeed cannot invent something new, instead, adopting as an externality a shared platform or system (the post, or email on the Internet, or a package delivery provider) already available (an externality) and indeed designed and engineered as a system, working at a higher scale, for a more general purpose than to join this journal with its authors and readers (namely to enable any such journal, and many others as well, and many activities and enterprises beyond journals). From this point of view, this journal's particular problem (as a "user", we might say) becomes only another instance of a more generalized problem – not maintaining a correspondence with readers and writers, but only an email system (or, before email existed, a postal system) among others.

It might be an interesting debate to discuss whether and in what respects the journals we produce today, with the support of electronic communications such as email and file exchange over the Internet, are "better" or "worse" than journals once produced on platforms we have long ago migrated away from. (A science of workflow might interestingly also be an archaeology of workflow.) Certainly the volume and rapidity of information exchange today is greater by orders of magnitude. There may also be shifts in who is able or permitted to participate, and for what presumed as well as actual purposes. However, we do not now prefer email, or our digital platforms of choice, to paper and postage, because they enable better work, so much as because the *scale* at which we now work – the number of partner exchanges we have, of what quality, and of what kinds of information codified in what forms – would simply not be achievable (much less sustainable) without the capabilities of the digital machine for information storage and manipulation.

We have seen a similar migration in the progression in "camera ready copy" for production of both print, and print surrogates, from literal image files, through printer instruction sets (such as PostScript ™) to today's PDF transmissions. All of these systems had to be engineered, but almost no one who adopted them paid much attention to the engineering itself. Each successive system merely met the need better than its predecessor. Today we have something far superior to what was ever possible without the networked exchange platform we now have (the Internet) and all the standards developed to support it. But no one exactly noticed much as our means of sending "pages" improved – as one subsystem replaced another. It simply happened.

Of course, it did not simply happen by itself, and the emergence of superior means (or at least, more capable means at scale) for maintaining business correspondence (email) or producing materials for the eyes of readers (camera-ready copy) reflected significant efforts by their developers and early proponents. The efforts were not made by the eventual users, however. Similarly, our users should not have to design their own technical solutions. A new platform emerges because we work at several levels of the system at once – and because we exploit emerging opportunities.

If our aim is to provide any "downstream user" with the kinds of capability and leverage one gets from external control, specification and testability of the kinds of regularities – at multiple levels of "semantics" – that can be usefully discovered or introduced into our data. We do this by working at different layers, providing users and indeed application developers not with solutions, but with the foundations and technical infrastructure on which their solutions can be constructed; but once that has been done, the solutions themselves "just work". In other words, what from one point of view, is an engineered solution, must be from another, just a better way to do the same thing.

Yet in a world where processes are already well defined and described, this is a challenge, since application

design must in some way come first – at least insofar as engineers must build to specific problems, not just general ones, to motivate the efforts. Solutions will emerge – someone will put effort into building them – if the base works well enough to enable and streamline these efforts. With a combination of good data, and good data description, we believe this is possible and necessary in a domain as complex and semantically rich as this one.

## How do we know we need (something like) OSCAL?

As pointed out in a 2019 paper for this conference, many or most of the functional requirements for data capture and relation (linking) that we face in systems security could be achieved, with only a bit of stretch, by extant markup technologies. The reason we need OSCAL is not because existing technologies including DITA, ISO/NISO STS, HTML microformats, XBRL, or even TEI, not capable of representing the data adequately (just to mention a few reasonable candidates[7]). Indeed if we assume that any of these has such capability, the question becomes why documents relating to systems security, and their exchange, remain firmly locked into proprietary technologies of production such as word processors and spreadsheets, given their well-understood limitations for systems working at scales beyond the enterprise. Why, in other words, has this migration not long ago happened?

One answer to this question is in plain view in the form of a common office document feature, namely templates, and the simple fact that an Office document (whether Microsoft Word or Excel, or a similar application on or off the web), for all its limitations, is the most flexible, powerful and accessible tool available (to one definition of "accessible") to a security professional, for data modeling. And data modeling – the definition, collection, management and deployment of structured, semantic data – is indeed at the core of their work. Available encoding standards all assume one thing: that the schema that adequately describes the document for its intended application, can be known ahead of time, indeed is not only known, but anticipated and accounted for by the general-purpose schema in question.

But every new structured document and every spreadsheet implies a model, and usually one (assuming a good designer) whose outlines are readily discernible to an informed reader. Indeed frequently, documents in use must conform to a "type" and follow rules for that type – evidence of the model again – with templates used as one (not the only) criterion for measuring conformance to the type (considered informally). Now, these new models are not made just for the enjoyment of it (although that could play a factor). On the contrary, we invest the effort because we can see the benefits (in higher quality, better and more throughput – that is, data processing capacity and capability) of doing so. The rules are not an impediment but a track to follow.

Many data professionals understand the shortcoming of office documents (considered as a genre) for truly widespread data, secure data exchange. But even they have no choice but to use them, since document templates are also what their own downstream users can use – while XML application stacks and libraries of stylesheets (or even the functional equivalent for JSON) are not.

Indeed it might be said that the essence of our problem is to make it possible for data professionals to do more than encode their information optimally for exchange (only) with their *immediate* partners – however necessary this is, and great the benefits of doing so. A workable standard for active information exchange is a *sine qua non*, but beyond it is another essential, since the requirements for exchange themselves in this domain are so local and so particular to processes, and defined and mandated at several levels at once.

This meant that whatever language we adopted or developed to address the needs, its own extensibility model would be crucial. (This is not an unfamiliar problem to the designers of documentary encoding standards.) Ideally, extensibility features would be free to users in the sense that no work in a schema or formal layer should be necessary for local applications to define and then enforce their own semantics (for example, by offering features enabling extension by restriction). But mechanisms for them to introduce such enforcement are also essential. We found a solution to this issue in our Metaschema technology (as described in the 2019 paper).

From this point of view, OSCAL (the Open Security Controls Assessment Language) should be regarded not as a solution, so much as a process for developing solutions. This process is technical and entails the definition and specification of information models as means to and end (successful, meaningful data exchange). But it is also very granular, and happens "on the ground". Like any documentary standard, OSCAL may be able to

provide for 80 % of what is needed for its “normal” cases, and what constitutes a normal case for it, as all technologies, will be a direct reflection of its capabilities. What is most important, however, is how OSCAL permits its users to deal with their 20 %.

It is not the aim of this paper to describe in any detail the OSCAL models or how they address requirements: a certain amount of background information on the project might be a prerequisite for much of what follows. It is too early to say whether and to what extent any adoption of OSCAL may change or improve the actual practice of security assurance. But it is not too early to reflect further on the challenges that could impede its success. And if the basic premise of OSCAL, like other markup technologies, is in declarative markup and descriptive encoding, it seems necessary to consider what problems or issues we might watch out for.

## The OSCAL Approach

This paper is for two audiences at once: information technologists who specialize in open data formats and the standards that sustain them; and systems and information security professionals who bring an understanding of the requirements of their domain, not necessarily deeply informed of available approaches or solutions (in the form of available technologies), but who bring an interest to this topic because they know or sense enough about it, to understand the significance of their impact. If anything, what makes our project interesting to the first of these audiences, is our relation with the second. Briefly, we are hoping to change the practice of systems security and its documentation, by presenting its stakeholders and practitioners with better ways of doing things. While we have some ideas (or we would not be making the attempt) of what these better ways look like, we must assume, however (quality is defined in context) that they are in a better position than we are to know what “better” will be. This means our primary challenge is to listen, with the goal of empowering them (users, stakeholders, the community, the market) to do what we would do for them, if we knew what they know.

In view of this, it may be worth considering briefly what OSCAL (the Open Security Controls Assessment Language) is, and is not. OSCAL is:

- A set of related and interlocking data models
- A data description language for a domain, and thus defined by that domain: systems security assurance and related documentary activities, as defined by IT (information technology) practice and statute.

As described in Piez 2019, these models are defined by a schema back end or *Metaschema* technology that permits us to provide support for these models in multiple syntaxes, specifically (to date) XML, JSON and YAML syntax.[8] This support takes the form of schemas for validation, conversion utilities and much else (documentation, starter stylesheets for designing representations, code generation, etc.). Together, these offer the platform for a stack of capabilities for data description, application and interchange. In this it is analogous to many other standard or common encoding technologies (XML-based and not), as they address their respective domains.

What then is OSCAL *not*?

- It is not a markup language.

  While OSCAL has an XML expression, and is designed for use in and with markup-based systems, it is also not a substitute for DITA, JATS/BITS, NISO STS, HTML[9] or any other extant markup technology, which are considered to be (from the OSCAL perspective) not alternatives (since they do not address the same set of functional requirements for data representation), but rather as complementary technologies and (as such) exploitable assets.
- OSCAL is also not an attempt to engineer a workflow or “solution” to the problem of data management in this complex domain. Rather, it is intended to provide the foundation or groundwork for the development of workflows and solutions.

  Again, existing workflows might be regarded as externalized assets and as opportunities, insofar as to the extent they can be “OSCALized” (enabled with and by OSCAL), they can work together with other systems more seamlessly, acquiring new capabilities via network effects.

So what about those functional requirements? The core concept is close to that of standards-based markup languages, albeit scoped within the particular domain of a specialized information set: we wish to enable better system security and security assurance by providing a foundation for rich (semantic) data exchange among partners and organizations. If we succeed, we will *lower the costs to organizations and users* of participating in such exchanges, by helping to *define and apply the rules* that enable it.

So how are the requirements we are addressing unlike requirements for publishing systems?

- When publishing for an audience of no more than three or four parties, requirements for production values are different, and economies of scale in production will not benefit in the same ways. Return on investment still comes from economies of scale, but not to the same (exponential) degree. At this time, achieving superior results, arguably, is as important as lowering costs. (This is not always true in publishing, which has been subject to economic stresses for much longer.)
- With respect to the data sets themselves, the granularity of description they require is (as compared to many applications of markup languages) relatively *rough*. This is reflected in the fact that OSCAL applications do not need much functionality at the word and phrase level – the data models is seeks to capture generally do not require it – and that when it comes to discursive contents ("prose"), it does not need much beyond some inline formatting plus a generalized insertion or transclusion mechanism (somewhat analogous to DITA **key/keyref**) working at the phrase level. Since these together can be accommodated using a near-subset of HTML or Markdown, the information can also (with some compromise) be constrained to representations that fit well within the limitations of JSON or similar object notations. In the terms I used in my Balisage 2018 paper Piez 2018, the data is higher on the "semantic stair".

  The flip side of this is that at a higher level of granularity – groupings of prose and structured data – the requirements for what might be called "hypertext" are comparatively intricate . Documents and their parts and components present complex interlinkings, both to one another and to similar or different parts of similar or different but related documents. For example, System Assessment Plans must make targeted references from their parts, to parts of system descriptions given in System Security Plans. Those references are "semantic" in the sense that they must be distinguished by type according to intended use and the kind of relations they encode; and when the links break, the documents break.
- Above all, OSCAL's users are different from the users of publishing systems or even from operators of documentary-production workflows.

  Most importantly, OSCAL's users will not only be CMEs (content matter experts) who use OSCAL-based systems to acquire and represent data to do their security assessment jobs, but also developers of systems and software to use it. While we do not expect that most OSCAL data will be published widely (the exception being canonical documentation such as the catalogs and baselines to which other OSCAL documents refer), we do expect it to be useful within organizations and between partners, in multiple unforeseeable ways. This means that developers need to be able to build to it.

  In order to win their support as well as maximize the chances for their success, we do not wish to constrain, any more than absolutely necessary, those developers with any encumbrances with respect to formats, software platform(s), or technical dependencies in general. Because we expect and rely on them to take us places we cannot go, we must trust them to use the means that seem most appropriate to them.

  In our experience, most dev/ops professionals become literate in multiple different formats for information interchange. However, it is also important to meet them where they are, and to enable them to use tools they know (while opening opportunities to use tools they do not yet know).
- This means that while we are free to define, demonstrate and promote models that enable functionality to be delivered, we are not free to stipulate that only XML may be used. Today, it is either JSON and other formats (including but not limited to XML), or JSON only.

  And indeed, since the information in our domain is not only documentary and not only suited to XML, having the capability to work in either format is a huge advantage. And designing from the start to be able to support and address either, also positions us over the longer term – as adding support for yet more alternatives (YAML, for example) becomes easier to do.

Notwithstanding these differences, both the complexity of the requirements, and the "documentary" and "fractal" nature of the data sets themselves – they exhibit regularities, but they are not entirely regular – necessitate the layered approach to systems design. As stated at the outset, we believe that a layered system that relies on declarative, descriptive encoding of data structures, with a separation from both underlying platforms (operating systems, storage media etc.) and from application logic, is the only practical approach to dealing with information management this complex. Indeed, the problem presented by systems security (planning, analysis, implementation, documentation, assessment) – whether considered as "micropublishing" or not – might be described as similar to the problem of designing robust, sustainable (platform-independent) publishing systems, except "on steroids": perhaps an order of magnitude more complex. The only way we have of managing that complexity is to factor it out into separate interrelated sets of requirements, which can be addressed separately as well as together.

If we have built and operated successful systems on that basis, the next question becomes what that experience teaches us.

## Applying the lessons: conclusions and expectations

To state that there is a way forward is not to say that it will be easy. Considering the lessons of XML in publishing, with its challenges, can help us reflect on the challenges we also face:

- **Your system is someone else's subsystem**

  This is even more true in the realm of RMF-based security assurance activities than it is in mainstream publishing. By design, an OSCAL document produced in and for one system, will be used, worked and integrated within another. An OSCAL system component description, for example, can be encoded once in one resource, then referenced by many systems plans that integrate that component. These documents will all be composed, produced and shared in different organizations (such as when the component in question is developed by one vendor and then used as a platform by another); and the links must hold together.

  One of the keys to scaling within the publishing domain is that an article, monograph or any published "artifact" is conceived as, in principle, a self-contained and self-sufficient entity, which can be written and produced for publication separately from others of its kind. Of course (as discussed above), this self-containment is partial and qualified, as much of the work of publishing is the integration of such an entity with other such entities within a larger structure – articles are integrated into a journal and even monographs are anchored into larger infrastructures for purposes of marketing, distribution, cataloging and so on. This is achieved through the application of two principles: (1) distinguishing "kinds" or classes of publication that can be treated alike within larger systems; and (2) associating metadata with each publication that can distinguish the instance among the members of the class.

  A primary goal of OSCAL is to begin to do this, and to provide a foundation for continuing to do so, within the domain of systems security documentation. Of course, the idea of kinds or classes of documents brings us to declarative markup.

- **Declarative markup adds value**

  The layering that is characteristic of systems based on declarative principles has been well explored at this conference and elsewhere. This layering enables separation of concerns between the production of data content (information sets), and its subsequent management, processing, rendering (presentation) and downstream application. When applied to publishing, this principle works.

  Again, however, the fact that there is a principle we can follow, does not make the design problem easy. In this case, distinguishing meaningful classes of information according to their nature, purposes and uses, depends on a clear and articulated sense of both commonalities across, and boundaries between different information sets, as well as their complex relations. Shared documentary structures that reappear throughout OSCAL's models reflect these commonalities; rules on their use reflect the boundaries. But both the shape of those structures and the rules applied to them, must make sense in terms of the data set as the practitioner sees it.

  And because our view of this is partial and evolving, we have also made efforts to enable the modeling to be agile and flexible and adaptive, most especially in the back end (Metaschema) technology we have developed to enable modeling across the gap between XML and object notations (see Piez 2019).

- **Data acquisition is hard**

  If only because it is so challenging in other domains, we should be ready to place special focus on developing means to convert relevant data sets into well-structured, well-described OSCAL.

  Multiple methods and approaches could be explored, using an "all of the above" strategy: structured editors; forms interfaces; semi-structured resources such as wikis or "issue" (ticket) systems; office document conversion pathways. Different methods or strategies may be appropriate for different parts of the system.

- **Activities are supported by incentive structures**

  The incentive structures within this domain are very different from publishing, and positive incentives are arguably scarce. Historically, activity related to systems security (beyond the functional minimum) has too often been a low priority, a kind of optional insurance policy for cases where the implicit security model of "trust your neighbors and leave the door unlocked" has failed. Similarly, the second-order benefits of well-documented systems security (exposure of latent issues; traceability; assurance; contingency planning) have been considered at best as "nice to haves". Without the heavy hand of regulation, security typically gets little if any attention from designers or developers until after a system is implemented and stakeholders are happy with its functionality and performance.

  In order for RMF-based activities to be fruitful, we need to keep incentive structures in mind, and look for opportunities to provide positive as well as negative incentives. To be sure, impediments must also be removed for positive incentives to come into play – thus for the OSCAL project we have done our best to address non-negotiable operational requirements (such as "XML versus JSON") in ways that make it possible for deployed systems to actually start to realize benefits in data interchange.

Noteworthy positive incentives should include, first and foremost, improved capabilities: more and better risk management at less expense. The more substantial positive incentives might take the form of better and more secure systems – that is, not only the documentation, but the systems themselves will be more secure, while also easier to develop, test, reuse and adapt in a "security first" mindset.

Additionally there are important secondary incentives, such as a more efficient use of time invested in assessment when the overhead of manual operations is reduced. Given that there is always more to assess, it is difficult to imagine how security assessments themselves can become cheaper. But with the aid of machining, and given better and more consistent, more easily consumed artifacts to represent the subjects of their assessment, one could expect assessments in general to be better, even to the point that "light touch" assessments (of well-documented, well-vetted systems) might be deemed to be adequate.

There are perhaps some further benefits of automation and automatability that might become incentives, to the extent that it can be recognized how achievable they are given appropriate investment. Much of the design of OSCAL is intended so that the considerable efforts of authors at lower layers – people who define and publish catalogs and baselines – can be better leveraged and exploited by the consumers of their information whether that be planners, assessors or others responsible for defining policy. This could become an incentive were it possible to monetize or otherwise feed back that benefit. (Pay a license to use an especially good baseline?) More likely, it becomes an incentive to the extent that such use and reuse of one's catalog (or baseline) is itself considered a criterion for success.

Finally, it is worth stressing that a significant factor in getting work done – to say nothing of good work – is inevitably in the imponderable aspects of pride and satisfaction with good work that good workers cultivate. It may be possible, by developing good tools, to build in "micro incentives" in the form of opportunities for good work, which serves as its own reward for those who see how consistency, transparency and integrity of the data they produce can contribute to the soundness of the system as a whole.

- **Quality is defined within context**

  With respect to security-related activities in general, or even RMF-based activities in particular, because operational context is hard to define and open-ended, no single solution will be a comprehensive solution. In the publishing domain, we have learned that the high degree of requirements for local adaptation and customizability has been critical to the success of the standard encoding formats. This is likely to be even more true for us.

  Tolerance of variation – and recognition of variation as a source of information – is an important characteristic of these systems. It being difficult to distinguish in general where variation is meaningful – where it is signal, and where it is noise – these systems need to be well defined, well managed and transparent, but also flexible and adaptable, with extension mechanisms that permit local adaptation without unnecessary "forking".

  Although it is outside the scope of this paper, the design of OSCAL's schemas and validation infrastructure permits addressing this set of issues in ways already familiar to designers and users of publishing systems: namely, by deploying not a single "one size fits all" validation regimen, but rather by supporting a layered or tiered approach, "mix and match". This permits organizations to define their own rules and rules sets and gain leverage over their own data, for their own (and partners') processes, even while they also conform to a more general set of rules shared by everyone.

- **Evolution works by little revolutions**

  Everything we have learned about the application of information technologies to publishing suggests that in this domain as well, progress towards better practices and more capable systems will be incremental before it is systemic. With a view to this likelihood, OSCAL aims to offer early rewards for users who can adopt it for solving problems, whatever those problems are, without imposing a requirement for any top-down overhaul. Even when OSCAL is never used at the core of a documentary system, it might be useful at its interfaces. And if it is useful at the edges of any system, it will eventually be useful at the core of others.

  Yet experience also suggests that developers and stakeholders must "get it", for progress to happen at all. There is no substitute for understanding, and thus for data transparency to the extent practical and possible. A commitment to open, non-proprietary declarative encoding – even within a secure operational context – is crucial, if only because a monoculture is not secure. But it is not only because of its long-term security, that the system must be open; it is because its success will depend above all on who understands it and how well, and how well they can adopt it for appropriate and intended use.

Within this context, one final principle might be recognized: *the platform is not the capability*. This might, indeed, be considered to be a core insight, in the sense that the entire enabling paradox of declarative markup is based on it – by defining the encoding in a way *independent of and abstracted from* its local application, we enable applications not only locally (any application must be local) but in general. While a technical platform (considered in the broadest sense) is necessary for a technical capability, this practical dependency is a

reflection of the fact that the logical dependency is the other way – unless it enables a meaningful capability, a platform or technical means remains inert and ineffective. A platform that offers no useful capability, will soon be abandoned. Conversely, while it is necessary for developing and demonstrating a capability, the very fact that a requirement can be described without commitment to a platform, is an indication that no platform or technical solution is a *sine qua non*. The different strengths of different technologies (whether XML/XDM, Javascript/JSON, comma-delimited values exported and imported into spreadsheets, or anything else) give them comparative advantages – and these can be exploited. Thus a platform that is developed to enable capabilities we already understand – and already have the means to accomplish – can also be a springboard.

## References


[declarative-bibliography] "Declarative Markup: An Annotated Bibliography" See https://markupdeclaration.org/resources/bibliography.html.

[rmf2018] Joint Task Force Transformation Initiative, "Risk management framework for information systems and organizations: a system life cycle approach for security and privacy," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-37r2, Dec. 2018. NIST.SP.800-37r2

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). 10.4242/BalisageVol17.Lubell01

Lubell, Joshua. "Using DITA to Create Security Configuration Checklists: A Case Study." 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). 10.4242/BalisageVol19.Lubell01.

Lubell, Joshua. "SCAP Composer: A DITA Open Toolkit Plug-in for Packaging Security Content." Presented at Balisage: The Markup Conference 2019, Washington, DC, July 30 - August 2, 2019. In Proceedings of Balisage: The Markup Conference 2019. Balisage Series on Markup Technologies, vol. 23 (2019). 10.4242/BalisageVol23.Lubell01.

[Lubell 2020] Lubell, Joshua. "A Document-based view of the Risk Management Framework." Forthcoming at Balisage: The Markup Conference 2020.

[McLuhan 1964] McLuhan, Marshall. *Understanding Media*. 1964. Cambridge and London: The MIT Press, 1994.

[a-130] Office of Management and Budget Circular A-130, *Managing Information as a Strategic Resource,* July 2016. https://www.whitehouse.gov/sites/whitehouse.gov/files/omb/circulars/A130/a130revised.pdf.

[OSCAL on the web] "OSCAL: the Open Security Controls Assessment Language." https://pages.nist.gov/OSCAL/ (accessed Mar. 24, 2020).

[Piez 2018] Piez, Wendell. "Fractal information is." 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.Piez01.

[Piez 2019] Piez, Wendell. "The Open Security Controls Assessment Language (OSCAL): schema and Metaschema." In *Proceedings of Balisage: The Markup Conference 2019*. Balisage Series on Markup Technologies, vol. 23 (2019). 10.4242/BalisageVol23.Piez01.

[Piez 2001] Piez, Wendell. "Beyond the Procedural vs Descriptive Distinction." Extreme Markup Languages 2001. Archived at http://wendellpiez.com/resources/publications/beyonddistinction.pdf

[Tillett 2004] Tillett, Barbara. "What is FRBR? A Conceptual Model for the Bibliographic Universe." Library of Congress Cataloging Distribution Service. Revised February 2004. Archived at https://www.loc.gov/cds/downloads/FRBR.PDF.

Walsh, Norman, and Bethan Tovey. "The Markup Declaration." 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). 10.4242/BalisageVol21.Tovey01

[xdm2017] XQuery and XPath Data Model 3.1. https://www.w3.org/TR/xpath-datamodel/


---

[1] SGML is "Standard Generalized Markup Language", ISO 8879:1986.

[2] LaTeX, a document processing system, is hosted at https://www.latex-project.org/.

[3] RMF is the *Risk Management Framework*, an approach to systems security management and documentation codified in NIST Special Publication (SP) 800-37. See rmf2018 and Lubell 2020.

[4] "Rheology" is a branch of physics. A science of workflow would be a branch of data science and cybernetics, applied at the level of the human organization, drawing (at least) from sociology, economics, general systems theory and operations research.

[5] FRBR is the Functional Requirements for Bibliographic Records, a model defining categories of description for bibliographical objects such as "books" and "articles" so that "the same" (book or article) can be distinguished and related systematically even across different variants or representations, including editions, translations, printings and copies. The "work" is the highest and most abstract category within FRBR. See Tillett 2004.

[6] As he writes in Understanding Media (McLuhan 1964 p. 8), "… the 'content' of any medium is always another medium. The content of writing is speech, just as the written word is the content of print, and print is the content of the telegraph."

[7] DITA: the Darwin Information Typing Architecture; ISO/NISO STS: the Standards Tag Suite; for HTML microformats see (for example) schema.org; for XBRL see https://www.xbrl.org/for TEI (Text Encoding Initiative) see https://tei-c.org/.

[8] YAML is "YAML Ain't Markup Language" (web site https://yaml.org/), a notation describing an abstract data structure amenable to processing in object-oriented languages. Its data model is an enhanced superset of the JSON object model; so by aligning with the requirements of JSON, an object model is thereby also expressible in YAML.

[9] JATS is the "Journal Article Tag Suite", an encoding standard hosted at the National Information Standards Organization (NISO), hosted at https://www.niso.org/standards-committees/jats. BITS is "Book Interchange Tag Set", a related encoding system hosted at the National Center for Biomatics Information, National Library of Medicine (NIH/NCBI); see https://jats.nlm.nih.gov/extensions/bits/. NISO STS is "Standards Tag Suite", a related encoding system designed specifically to support the publication and maintenance of technical standards documents: see https://www.niso.org/standards-committees/sts.

Approximate word count: 13125. 2 figures.

***Balisage:*** ***The Markup Conference***