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***