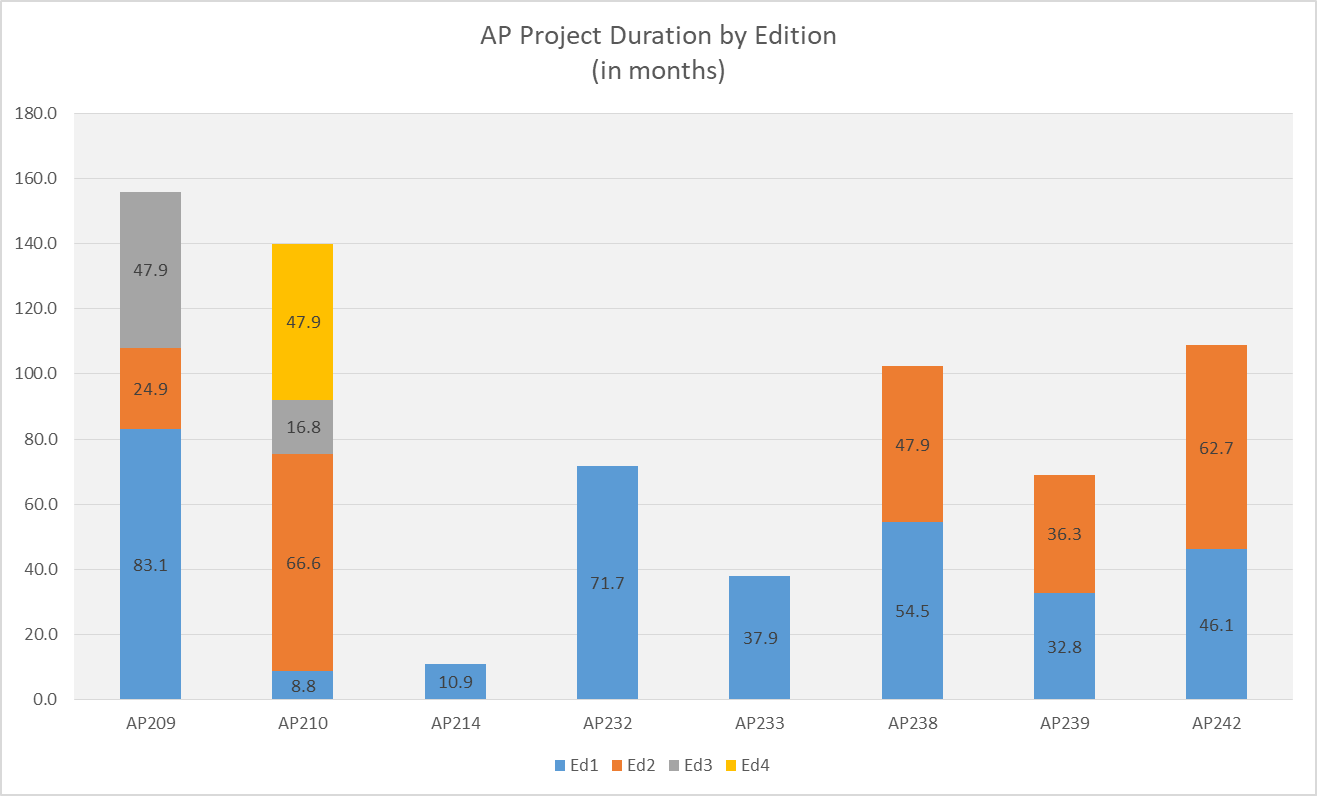
**Current State**

"We are implementing features and products and using technology that were not invented 18 months ago. No longer can we afford these large monolithic programs that go on for two to three years (24-36 months)." -Bronwyn Clere, Executive Director for Capital Planning & Delivery, at Telstra Corporation. An analysis of ISO project metrics on a sample across 8 standards and 16 edition publishs revealed the current average project duration is 43.5 months for the release of an edition of a standard. This calculates the time from stage 10.99 of New Project Approval to stage 60.60 International Standard Published. Additional analysis reveals this is an average project length growth of 4.5 months from edition 1 lifecycle lengths to that of edition 2 lifecycles.



*Figure 1: Average project duration by edition (data based on ISO.org project metrics). Note these metrics are based on a status range from stage 10.99 (new project approval) to 60.60 (international standard published). This is a truncated view and does not include the white paper authoring process which can add as much as an additional year to project duration.*

This trend of project length growth for standards is at odds with the industry’s need for faster releases of incremental functionality. ISO’s 2016-2020 strategic directions include six tenants two of which are (1) “Develop high-quality standards through ISO's global membership”, by ensuring we effectively (2)“ Engage stakeholders and partners ”. In order to support industry demand and maximize the benefit of standards the quality and length of development of standards must be addressed.

***Development Lifecycle Management***

There are two primary contributing factors causing an extended development cycle:

1. Failure to gain consensus [International Agreements]
2. (2) Resource commitment/management.

In the course of STEP development once a new work item is approved or a defect raised, most teams use a system such as Bugzilla for coordination. Bugzilla was originally designed as a general purpose bug tacking and testing tool (this is from wiki, find a better source) however in STEP development it has been stretch beyond that scope. In existing STEP development application it is used as a requirements management, collaboration /consensus tracking, issue management, and task management system in addition to for version control of source models. This leads to much confusion, maintenance, and misuse of Bugzilla. The result is extended development cycles and many times rework during the committee stage 20.

Resource commitment for the performance of development work is also a primary obstacle. Factors that affect commitment of resources include:

1. Volunteer staff
2. Lack of developer resources (which are shared by multiple ISO projects)
3. Virtual distributed team

***Quality Management***

Another issue facing the current development lifecycle is that of quality/completeness concerns of the published standards. CAx-IF, which is a joint testing forum between AFNet, PDES, Inc. and prostep ivip tasked with testing CAx STEP translators quality, has continually reported issues of implement-ability of the standards. These issues fall into three areas and can be categorized as data quality issues, incomplete solutions, and overtly complexities/non implementable solutions. Metrics from Bugzilla show 420 bugs ave been reported in these three categories by the CAx-IF against STEP to date. 

While measures have been taken combat complexity and streamline solutions, such as the transition from monolithic to modular architecture, the problem still persists.

There are two primary root causes contributing to the quality/completeness issues of standards:

1. Lack of knowledge
2. Lack of adequate toolchain

A robust knowledge of the STEP data model is necessary to ensure enhancements and defect resolutions have a complete end to end solution. The integrated nature of the elements of the STEP data model, ARMS, MIMs, and IRs, mandate that parallel changes take place in each of these elements to ensure continuity of the model. A lack of understanding of integration points leads to (1) inelegant solutions and (2) data quality issues. While quality checks exist to catch data quality issues these checks are not infallible. Other issues, such as the elegant or clumsy nature of solution concepts, require analysis by human interpretation and rely solely on the knowledge level of the human resource performing the work. Understanding of the development process and toolset are also factors that affect code quality. Results from a survey of the standards development team reports that 89% of developers incurred some amount of rework during their last publication project due to lack of knowledge of the data model, development process, and/or tool-chian. Of those 7% reported more than 50% of rework was required. For a view of the survey results in their entirety reference appendix B. See the below figure for survey results of perceived knowledge level by tool-chain application. ****

The second primary root causes contributing to quality/completeness concerns is an inadeqauate tool-chain.The current tool-chain for model based standards is fragmented which is the result of it’s incremental development over the last 30 years to adopt fit for use tools to specific tasks. While this has allowed development to go forward it has also resulted in a disjointed tool-chain which relies heavily on process controls and manual integration to achieve end to end project management and development tasks. This leads to a temperamental development process, poor collaboration and version control, and manually introduced errors.

References

<https://ieeexplore.ieee.org/abstract/document/6475408>

<https://www.iso.org/files/live/sites/isoorg/files/archive/pdf/en/iso_strategy_2016-2020.pdf>

<https://www.iso.org/files/live/sites/isoorg/files/developing_standards/docs/en/Target_date_planner_4_ISO_standards_development_tracks_2017.pdf>

<https://www.cax-if.org/>

<https://www.cio.com/article/3304276/speed-adaptation-and-the-pace-of-change.html>