# Einleitung

Hi Grok! Ich würde gern da weitermachen, wo ich zuletzt aufgehört hatte mit Dir zu arbeiten. Es folgt der Verlauf unseres letzten Gesprächs (ich hoffe, das klappt):Ich arbeite an meiner Promotion, bin im Bereich Bauinformatik unterwegs und entwickle derzeit eine Ontologie, die OCCP. Diese möchte ich nutzen, um mein Konzept des chronologischen Modells (Modell meine im Kontext von BIM und IFC), kurz cMod umzusetzen. Dazu möchte ich ein Minimalbeispiel erstellen, das unterschiedliche Zustände eines Gebäudes darstellen kann. Dazu benötige ich wiederum eine Software, die den Arbeitstitel cMM (kurz für cMod-Manager) trägt, mit deren Hilfe ich ein cMod erstellen, laden, bearbeiten, speichern und darstellen können möchte. Kannst Du mir helfen, dieses Projekt umzusetzen? Nachfolgend sende ich Dir im Anhang meine bisher dazu erstellten Paper bzw. ext. Abstract, damit Du thematisch besser abgeholt wirst. Bevor wir in die Entwicklung einsteigen, werde ich Dir einen fundierten Überblick über den aktuellen Entwicklungsstand geben. Anschließend können wir uns gemeinsam überlegen, wie wir am besten weiter vorgehen.

## EC3 Paper cMod 2023

Hier also der Textinhalt meines ersten Papers zu diesem Thema:Abstract Model-based planning requires an extensive amount of information about the environment, inventory and requirements of the planned object. Thereby, temporal information is important to be considered for planning, construction and maintenance, but in current approaches temporal information is only defined through non-semantic datasets, which lack contextual meaning. To semantify time-data that is used in Building Information Modeling (BIM) projects, the authors propose the concept of combining existing models using Information Containers for linked Document Delivery (ICDD) and adding an Ontology for Chronological Construction Processes (OCCP) to record and implement structured temporal information of the entire life cycle and use it from the start of planning until deconstruction and even beyond. Introduction Obtaining all relevant information is a recurring challenge in planning projects. Difficulty and effort scale with the increasing complexity of the project, e.g., with the number of people involved or the number of disciplines to be considered. The infrastructure sector places special demands on planners due to its public relevance and the resulting large number of stakeholders involved. To reduce the planning effort, the efficiency of information procurement must be optimal, because it is the only way to avoid iteration loops that usually result from poor data bases or poor communication. One approach for creating an optimal information base is the development of comprehensive As-Built databases in the form of digital twins of the so-called As-Built or As-Is models in the context of BIM. A modularized structure of individual, self-contained models and the subsequent, georeferenced linking of these models enables the step-by-step construction of large models of high complexity. Nevertheless, even a ‘perfect’ digital twin still harbors the risk of planning errors, for example when unknown and therefore unmodeled old structures (e.g., old foundations or pipelines of a preexisting construction left remaining in the ground) only come to light during construction. Working with models for planning and operation, a lot of temporal information can be added to models, but so far this information is not semantically structured. Therefore, interpretation of these data is either processed manually by human experts, which could result in a time-consuming and error prone task or parsed by software tools that need to know the exact data structure in which the time information is defined. The latter solution requires continuous updating of the parser whenever changes are made to the interpreted data structure. Also, in the course of planning, several versions are often created in order to record different variants of planning and construction states. When As-Is models are used in operation, several versions of a structure or area may also exist, since the model is used as an As-Built document and is continuously updated. Maintaining multiple versions may result in greater storage requirements and can lead to problems identifying specific conditions at a given point in time. This problem can be addressed by logging versions or using version control systems, but the more different points in time or versions are archived, the larger the number of versions that need to be managed. In the context of planning, versioning is useful because all versions that are no longer needed can be deleted once planning is complete. In relation to As-Is models, versioning is not optimal for the above reasons. The introduction of chronological models provides an approach to counter the described problems by reducing the number of versions needed to differentiate between varying states of constructions and components. In chronological models, temporally relevant information is added to the model with the help of an ontology, making versioning superfluous and semantically enriched. At the same time, an unambiguous mapping of construction model states at arbitrary points in time is enabled and redundant data storage is favored, which can ensure a comprehensive provision of relevant planning information. State of the art Implementation of temporal information Possibilities for defining time-dependent information already exist in Industry Foundation Classes (IFC) of ISO 16739-1, which is an open BIM standard. A specific entity for defining time explicitly is provided in the form of IfcTimeSeries and its related subclasses. Thereby, an instance of IfcTimeSeries defines a time period that is detailed through multiple points in time. However, IfcTimeSeries is intended to be linked only with external references outside of the IFC model that could be referenced via an Unique Resource Identifier (URI), e.g., dictionaries or documents. Furthermore, IfcTimeSeries just provides functionality for defining raw time data, but not assigning this data to a semantic meaning, e.g., the lifetime of a building or the point in time of its deconstruction. Therefore, meaningful temporal information is defined as attributes of other IFC entities. For instance, process related entities that are subtypes of IfcProcess, such as IfcEvent or IfcTask, possess attributes for explicitly binding a point of time to an event or task. By utilizing these entities, it is possible to structure various processes in the construction life cycle, such as the design or inspection and annotate them further information. However, since no entities or attributes for specific life cycle processes are supported by IFC, e.g., the submission and review of a design document, the interpretation of this information must be processed either manually by humans or by software applications that are aware of the concrete meaning of the non-standardized annotations. An approach that proposes utilizing IfcProcess and its subtypes for defining time-related information about structural damage inspections has been developed by Artus (2021). In his publication Artus (2021) also discusses the advantages and drawbacks of using the existing IFC without extensions for describing time information. The main advantage of this approach is that the model would be directly readable by many IFC-supporting applications. However, the main drawback would be that some entities would not be used in the semantically correct context. To overcome these drawbacks, IFC extensions have been proposed, e.g., by Tanaka et al. (2018) for describing component degradation over time or by Eftekharirad et al. (2018) to develop an IFC-based system for fire emergency real-time management. In this regard, new IFC entities were developed for describing a certain object or behavior in a specific point in time. Similarily to IfcEvent or IfcTask, the definition of time is assigned to the entity instances via a corresponding attribute. Nonetheless, a processing of these extensions in publicly available BIM applications is currently not supported. A valid option for describing time semantically, so that software applications could interpret this information in runtime without any prior knowledge about it would be the utilization of Semantic Web Technologies to develop ontologies. Through utilizing these technologies, it is possible to represent information in structured graphs and efficiently integrate heterogeneous data from various information sources as well as performing complex search queries on these sources for an enhanced information management and interoperability in the Architecture Engineering and Construction (AEC) industries through the whole construction life cycle (Pauwels et al., 2017). In addition, the underlying logical foundations of the languages used by Semantic Web Technologies such as the Web Ontology Language (OWL) or the Shapes Constraint Language (SHACL) allow for logical inferences and proofs of asserted information. Since the knowledge is linked to the corresponding information inside the ontology, it could be exchanged between different software applications without bilaterally adjusting the applications to new types of information. Instead, the new information is interpreted in runtime by utilizing a compatible reasoning engine. Furthermore, ifcOWL, the OWL representation of IFC, allows the use of Semantic Web Technologies on an IFC-based model (Pauwels & Terkaj, 2019). An established ontology for defining time concepts is the OWL-Time ontology (Cox & Little, 2022). It supports the definition of points in time through the class time:Instant as well as for time intervals via the class time:Interval. Furthermore, multiple object properties for structuring time sequences as well as aggregating time objects in intervals are provided. Additionally, the declaration of time values is possible through various data properties. However, the OWL-Time ontology does only enable the definition of general time concepts. Consequently, domain specific time concepts, especially those used in AEC, are missing. An alternative solution has been developed by Milea et al. (2009) through proposing a language extension of OWL called tOWL, which provides additional language constructs for defining time. However, ontologies formalized in tOWL require a specific reasoning engine different from common OWL reasoners, which prevents its usage in existing OWL applications. Several approaches reuse concepts of the OWL-Time ontology for AEC-related knowledge representations. For example, Mignard & Nicolle (2014) developed an ontology for merging BIM and GIS information in which the temporal concepts were based on the OWL-Time ontology. Iadanza et al. (2019) used the OWL-Time ontology to describe model phases through time and represent the building evolution. Furthermore, Zheng et al. (2021) developed an approach for representing the construction workflow utilizing the OWL-Time ontology. Similar to the process definitions used in IFC, process classes are defined in an OWL ontology. By using the existing concepts of OWL-Time time components are added to process representations. Linking data with information containers One way to integrate an ontology in a model is the use of information containers. To avoid interoperability dilemmas and support collaboration, the information containers are one of the recently developed approaches proposed for the management of heterogeneous and distributed building models. The ISO 21597 series has been developed to fulfil a requirement for multiple documents to be delivered as one information package within the construction industry. The international standard ISO 21597-1:2020 provides a framework for the creation, management, and delivery of linked documents. Over the last few years many studies were conducted to investigate the implementation of the ICDD. Hagedorn et al. (2022) proposed the concept of the Toolchain Framework to facilitate the modeling of project-specific workflows by linking individual software tools based on a standardized process notation to enable seamless information exchange between applications that integrate an openCDE-compliant web interface. Just recently the author proposed a BIM-based solution for Infrastructure Asset Management System (AMS) for road owners. Considering the requirements of stakeholders across domains in the operational phase, the proposed approach provides asset managers with a strategy for the dynamic use of Information Containers (Hagedorn et al., 2023). Moreover, Liu et al. (2021) developed the Building Concrete Monitoring Ontology (BCOM) based on the ICDD. It allows IFC-based bridge models to be configured with properties about concrete works that can be processed by predefined queries in asset management software. An approach was developed by Hamdan et al. (2021) in which an IFC model, representing an existing bridge, was linked with ontologies that semantically represent the construction and affecting structural damage as well as other related data, e.g., photos, protocols or structural analysis models. Thereby the models and links were stored in an ICDD. Ye and König (2021) presented a framework for automated billing by combining the BIM Contract Container (BCC) with Smart Contracts using Blockchain technology and the ICDD. Furthermore, recent research by Werbrouck et al. (2022) proposed a Linked Building Data (LBD) server that can link heterogeneous linked building data in a Federated CDE by combining the Solid initiative for web decentralization with the ICDD standard. Going beyond the ICDD implementation, Al-Sadoon et al. (2022) proposed an ontology-based extension that enables allocating of multiple values for elements in the IFC files to provide dynamic building model for simulation tools using the ICDD. The concept of chronological models To create models with structured temporal information - not only for models of entire buildings, but for models of each delimitable component of a structure (in logically meaningful gradation, i.e., the largest possible independent component group) - the implementation of Chronological Models (cMod) is proposed. The concept of cMod is based on the use of IFC-based models of components that are linked inside an ICDD with other data and other ontologies, such as the Building Topology Ontology (BOT) by Rasmussen et al. (2020) or the Bridge Topology Ontology (BROT) by Hamdan et al. (2020). The temporal information is connected to the IFC model via ICDD by adding the proposed Ontology for Chronological Construction Processes (OCCP), containing the chronological structure shown in Figure 1 and specific, component related time information. This information provides statements about: A - relevant times of planning of the component (e.g., start of planning, data procurement, submission for review, changes (resulting in multiple submissions for review)) B - relevant times of review (at least the time of acceptance and rejection – the latter causes another planning cycle, adding more changes (that are tracked again with the according temporal information as described in ‘A’)) C - relevant times of construction (start of execution, completion, issuance of notice of defects, completion of defect rectification, acceptance) D - the times of commissioning of the component (if different from completion) and decommissioning (or start and end of use, multiple entries possible (e.g., to note temporary closures)) E - the dates of the beginning and end of the warranty period F - the times of beginning and end of the design life (multiple entries possible, e.g., due to maintenance or repair works) G - the scheduled and actual times of inspections and the length of regular inspection intervals (the cycles must be specified) H - the times of special events (severe weather events, accidents, etc.) K - the times of detection of damage and the start and completion of repairs (to keep track of how long a damage existed) L – the times of repair and maintenance works (repair work is to be treated as a new component, for which the temporal information is given in categories A to M as needed.). Thus, if necessary, temporal information of other components with reference to the repair work is updated or supplemented with corresponding new values (acceptance, new predicted service life, warranty and so on)) M – the times of beginning and end of deconstructionFigure 1: Temporal structure of the OCCP The structure shown in Figure 1 is a suggestion for the sensible and practicable recording of time-specific component information. An important prerequisite for the use of temporal information is the agreement in the industry regarding the relevant temporal information. The proposed structure is based on the life cycle of a construction and divides the processes at common milestones. The process in the shown concept is kept simple and general to promote transferability and interoperability for all AEC disciplines. Further specifications, additions and smaller process steps can be developed individually according to requirements and established in exchange with experts. A cMod differs fundamentally from the versioning of (partial) models, as it offers far more flexibility due to its component reference and makes multiple storage of individual elements in several versions of a model obsolete. The state of a structure can be mapped unambiguously by specifying the point in time under consideration. In contrast to versioning, each component would be modeled once and supplemented with the chronological temporal information. In the case of a repair measure, the component model must be updated by separating the remaining part from the damaged part that is to be repaired. The damaged section is treated as a new cMod that goes through the phases from planning until construction and is then to be modeled as built and spatially connected to the remaining, undamaged, and unchanged rest of the component. At the end of the life cycle, the deconstruction date must be noted, but the model must not be deleted. This ensures that all information within the cMod is preserved and can be made available for planning purposes in the future. Starting from a database in which all components including the respective information and their references to other components are stored, the Linked Data approach is used to generate the overall model of a construction from many individual modules. By specifying a certain area, it is possible to limit the components that must be considered for the further query (e.g., building X, 1st floor). By specifying the time of viewing, the elements to be displayed for the selected time are filtered out from all elements in the viewed area by evaluating the chronological component information and are then activated for display. Ontological basis within ICDD for the implementation of temporal information For the structuring and later linking of the temporal features with models or model components, the OCCP is proposed, which extends existing classes and object properties of the before mentioned OWL-Time ontology (Cox & Little, 2022). Figure 2 shows the general class structure of the OCCP. The right side of Figure 3 shows the structure of the ICDD used to link the OCCP with the IFC models of a construction and its components, thus creating a cMod. The left side of Figure 3 shows an individual of a construction (IND:ExampleConstruction) with several components linked with BOT to connect and describes the topological relation between the construction and its components. The temporal information of each of these individuals is added using the OCCP and linked to the corresponding IFC model with an ICDD linkset. The IFC sub-models of the construction components are linked to the IFC model of the construction via ICDD linkset. In the OCCP the classes time:Instant and time:Interval of the OWL-Time ontology are extended. Both classes are subclasses of time:TemporalEntity, which is a general class used for describing temporal concepts. Thereby, time:Instant describes a specific point in time and thus a temporal entity with zero extent or duration. Contrary to this, time:Interval describes a temporal entity with an extent or duration (Cox & Little, 2022). With the OCCP the temporal entity occp:Phase is introduced as a subclass of time:Interval. Instances of occp:Phase represent major life cycle stages of a building or construction element and are used to categorize its temporal information. Furthermore, to differ between standard intervals and cycles that are periods of time with several cascading intervals within, a new type of interval called occp:Cycle is introduced, which is also a subclass of time:Interval. This class is used to better describe iterating processes like regular inspections and their intervals. For instances of occp:Cycle a data property called hasCycleNumber can be utilized for defining the number of iterations that need to be processed by a cycle. Additionally, a new subclass of time:Instant called occp:Transition is provided in the OCCP, which is used for representing milestones between the life cycle phases of constructions. For managing the life cycle phases and temporal entities that are associated with them, additional object properties have been added, which are shown in Figure 4. The chronological order of the phases is defined using occp:phase\_after and occp:phase\_before, which are subproperties of time:after and time:before. Each phase has a beginning and an end, both of which are defined by referencing a corresponding time instant using the OWL-Time ontology’s object properties time:hasBeginning and time:hasEnd. Additional time instants within the proposed ontology mark typical key points within phases. An important part of the OCCP is the semantification of AEC-related time data. For this reason, various domain-specific subclasses of occp:Phase and time:Instant have been developed, based on the concept of chronological models discussed in the previous chapter. In this regard, subclasses of time:Instant that belong to a certain life cycle phase, are categorized in corresponding superclasses, such as occp:PhaseA\_Instants or occp:PhaseB\_Instants and so on. Additionally, subclasses of occp:Transition provide more specific meanings to milestones in between life cycle phases. Furthermore, additional semantic information of a construction or component can be added to a model (e.g., topological information via BOT) and thereby extend the possibilities of information retrieval through queries. In their current states both the cMod and the OCCP are concepts and we envisage to work out test cases in future research work. Example of ontology application Figure 5 demonstrates the semantic structure of temporal information within a cMod using the OCCP, but for demonstration purposes, not all existing connections are shown. A construction component is represented by the individual IND:ConstructionComponent and it is linked via occp:hasPhase to two individuals of phases. While IND:PhA\_Planning is an instance of occp:PhaseA\_Planning, IND:PhB\_Review is an instance of occp:PhaseB\_Review, both of which are subclasses of occp:Phase, which in turn is a subclass of the time:interval. Linked to the component are several individuals, all of them marked with an “A” that either belong to the planning and the review phase. The links are established using the object property time:hasTime with the according individual, e.g., IND:PhA\_Start and IND:PhA\_Measurement. Block 2 in Figure 5 shows the semantic connection of the A-individuals. They all have the object property of the according class within the OCCP, e.g., IND:PhA\_Start is an instance of occp:BeginningOfPlanning, and IND:PhA\_Measurement is an instance of occp:DataProcurement and so on. The rest of the semantic chain is the same for all A-individuals, as they are classified as occp:PhaseA\_Instant or occp:PhaseB\_Instant (depending on the phase they belong to), which is a subclass of time:Instant. The chronological structure is established by defining the succession of time instances and phases using object properties, such as time:after or time:hasBeginning. The beginning of the planning phase is marked by the start of planning, in the OCCP this connection is made by using time:hasBeginning and referencing IND:PhA\_Start. Analog, the end of planning is defined by linking IND:PhA\_Planning using time:hasEnd and referencing IND:PhA\_SubmissionToReview. The order of instances within a phase is established with the object property time:after, e.g., IND:PhA\_Measurement is defined as time:after IND:PhA\_Start and so on. The same principle is used to describe IND:PhB\_Review (and all following phases - not shown for simplicities sake) in order to link and structure the according individuals of the according instances. The planning phase ends with the submission of the component’s plan. The case of rejection of a plan and the resulting iteration can also be described using the OCCP, but is not shown in this example to keep Figure 5 readable. The review phase ends with the approval of the plan and IND:ConstructionComponent is linked to the transition instant IND:T\_PlanningCompleted, which has the rdf:type of occp:CompletionOfPlanning. All individuals in this example, aside from the component’s and the phases’ individuals, receive an individual temporal information using the data property time:inXSDDate. Block 1 in Figure 5 shows the temporal order of the planning and review phase and the transition marking the completion of planning. Using this semantic structure for temporal information provided by the concept of cMod for each component of a construction enables a variety of options. The planning progress is tracked for each component individually and delays are potentially more obvious (because they are queryable), including the identification of the component causing the delay. This and all other temporal information can be stored within the OCCP, linked to the IFC-model via ICDD and therefore accessible for future planners, working with the model as a basis for the new planning project. All data used as a planning basis, like the measurement of the land or geological data, is linked to the construction model and the time of procurement is stored in the OCCP of the construction model. That way the actuality of all linked data can easily be checked and in case of a certain demanded actuality, the latest point in time to acquire a new set of data can be identified and used to remind the involved personnel in time. Also, the exact point in time and order of events can be of interest to answer questions of legal responsibility and it often comes down to the question, who had which information and when. The temporal aspect can be answered using a cMod. Potentials and applications of chronological models The concept of cMod that is proposed in this paper could be implemented in software applications and enhanced with further reasoning, querying and validation functionalities in the future. Provided that this chronological component and model information has been implemented, it is possible for planners to obtain an overview of the old construction development in an area relevant to planning, e.g., to view old construction conditions or to identify construction remnants that may possibly collide with the planning object. By specifying the area of a model to be displayed and a point in time, which could be the current state or any state in the chronological model’s past, provided there is temporal information in the point in time of interest, the combination of partial models that existed at the specified point in time can be displayed via a query of the chronological module information. By concatenating the chronological information and querying time periods (e.g., from a certain point in time until today), building states could be displayed in arbitrary increments (1 week, 1 month, etc.) and thus extensive information regarding the building states, building sequence and any backlogs could be visually displayed in one cMod. By introducing time as a linear dimension within cMods, the space related freedom from contradiction or collision is eliminated and replaced by the freedom from contradiction in time. The result is that - looking at a fixed part of a model over a certain time period - several components (e.g., an old and a new window) can be at the same position within a building model but cannot be at the same position at the same time. This enables the continuous use of existing models and thereby reduces the time and effort needed to collect all necessary information for planning the next construction, as it will already be integrated in the OCCP within the cMod. This also implies that a versioning of models is no longer necessary, because through the difference in their temporal information, both the old and the new component can exist within the same model and in the same place. For the actual versioning process, existing approaches could be considered such as the Ontology for Property Management (OPM) by Rasmussen et al. (2018), which could enhance the querying time compared to timestamps. Further potential applications arise from the operation of buildings and constructions, as administrations have a quick overview of warranty periods that can be used. An automated query of the remaining warranty periods can generate reminders with sufficient time to still be able to perform an investigation before the warranty expires. Another area of application is the evaluation of types of construction, materials or construction elements regarding their suitability for the respective application by comparing the design service life to the actual service life or by monitoring the needs of related companies for repair measures. The effectiveness of repair measures can also be evaluated by following up the service life. The chronological model could be validated against national standards, by utilizing SHACL. For instance, certain nation-specific process sequences are defined in corresponding standards and must be executed in a specific order for which SHACL shapes could warrant the correctness of their implementation in the model. Furthermore, the implementation of additional rules that could be defined in rule languages like Jena Rules, SHACL or even OWL itself, could allow for a more automatized logic-based application of the cMod. Thus, implicit knowledge, subsequent phases or required measurements or processes could be inferred through reasoning an ontology using the OCCP. Since the current proposed concept just defines the base taxonomy based on existing expert knowledge, the need for future features needs to be identified, e.g., through developing competency questions or use cases via expert interviews. Based on the identified application needs, additional rules and queries could be developed and implemented in OCCP-compatible software applications. Outlook and conclusion To make the advantages of working with cMods available for all stakeholders involved in the lifecycle of a construction, the concept of cMod and OCCP would first have to be implemented in the software used by these stakeholders, covering all aspects of time relevant information across all phases. After the implementation, the access, usage, and application of the time ontology related features and information should be intuitive and as easy as the spatial assignment of model information. Temporal collisions and logical contradictions must be either prevented or indicated by the software. An integrated query function should use the temporal information of cMods and make it easily accessible, e.g., by presenting a selection of information depending on the temporal information given by the query. Input data for the queries must always be a point in time or a time span and a specific location (three-dimensional delimitation of the space or specific components) to be considered. In case of a fixed spatial viewing area, a timeline function could be implemented in the software, whose extension maximum results from the earliest and last entry of temporal information. Analogous to the playtime display in music and video players, a corresponding slider could allow the control of the considered point in time and thus visualize all states of the considered object over the entire period. The corresponding model information is also displayed or retrievable depending on the considered point in time. A potential loss of temporal information could occur when working with the IFC model of a cMod without using OCCP. To address this problem, one solution could be the use an API to implement an export function for temporal information. Further research should be done to

* investigate possibilities to securely store and manage temporal information to guarantee manipulation security and to answer legal questions respectively avoiding legal problems using the temporal information (e.g., by using automatically generated time stamps that are permanently unchangeable or by integrating the block chain technology).
* ensure that temporal information is free of contradictions (e.g., plausibility checks could be established by utilizing description logic in OWL).

develop a rights management for the entry of temporal information (e.g., by assigning processor rights in combination with the 4-eyes principle (or more), whereby the authorized persons must be determined project-specifically and by mutual agreement (e.g., one representative for each client, contractor, and an independent expert / construction supervisor / BIM manager)). - establish practical and meaningful rules for the storage of model information to create historical models from the past and for the future (e.g., by working out an agreement for indefinitely storing a defined minimum of model information). - create an API to import and export temporal information stored within cMods to improve interoperability. The use of chronological models, created by the permanent integration of semantically structured temporal information in models of constructions and their components using ICDD and the OCCP, reduces the need of versioning due to the possibility of spatial coexistence of multiple components that is made possible by the temporal differentiation. The continuous use of existing cMods can prevent the loss of information and has the potential to save time for planners, as the time needed for data collection for new planning projects is minimized. With the adaption of OCCP, all temporal information can be semantically stored and used for queries, project managing purposes, and to help solve legal issues. The authors share the concepts of the cMod and the OCCP at an early stage of development to discuss it with the AEC community and to receive feedback before making the first version of the OCCP and a minimal example of a cMod public. References Al-Sadoon, N., Katranuschkov, P. & Scherer, R. J. (2022) Extending ICDD Implementation to a Dynamic Multimodel Framework. In: Proceedings of the Conference ECPMM (doi.org/10.1201/9781003354222-15). Al-Sadoon, N., Scherer, R. J. &Menzel, K. (2023) From Static to Dynamic Information Containers. In: Proceedings of the Conference EC3 2023 (in press). Artus, M. (2021) Modeling Physical Damage Information at Concrete Bridges Considering Time Aspects. In: 32. Forum Bauinformatik 2021. Darmstadt, Germany. Cox, S. & Little, C. (2022) Time Ontology in OWL. <https://www.w3.org/TR/owl-time/> . Accessed date: 25th April 2023. Eftekharirad, R., Nik-Bakht, M. & Hammad, A. (2018) Extending IFC for Fire Emergency Real-Time Management Using Sensors and Occupant Information. In: 35th International Symposium on Automation and Robotics in Construction (ISARC) 2018. Berlin, Germany. Hagedorn, P., Block, M., Zentgraf, S., Sigalov, K. & König, M. (2022) Toolchains for Interoperable BIM Workflows in a Web-Based Integration Platform. Applied Sciences. 12(12):5959. Hagedorn, P., Liu, L., König, M., Hajdin, R., Blumenfeld, T., Stöckner, M., Billmaier, M., Grossauer, K. & Gavin, K., (2023) BIM-Enabled Infrastructure Asset Management Using Information Containers and Semantic Web. Journal of Computing in Civil Engineering, 37(1), p.04022041. Hamdan, A.H., Taraben, J., Helmrich, M., Mansperger, T., Morgenthal, G. & Scherer, R. J. (2021) A semantic modeling approach for the automated detection and interpretation of structural damage. Automation in Construction, 128, p.103739. Hamdan, A. H. & Scherer, R. J. (2020) Integration of BIM-related bridge information in an ontological knowledgebase. In: Proceedings of the 8th Linked Data in Architecture and Construction Workshop (LDAC)Iadanza, E., Maietti, F., Ziri, A.E., Di Giulio, R., Medici, M., Ferrari, F., Bonsma, P. & Turillazzi, B. (2019) Semantic web technologies meet BIM for accessing and understanding cultural heritage. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-2/W9 (2019), pp. 381-388 Liu, L., Hagedorn, P. & König, M. (2021) An ontology integrating as-built information for infrastructure asset management using BIM and semantic web. In: Proceedings of the 2021 International Conference on Computing in Civil and Building Engineering, 99-106. Mignard, C. & Nicolle, C. (2014) Merging BIM and GIS using ontologies application to urban facility management in ACTIVe3D. In: Computers in Industry 65 (2014), pp. 1276-1290. Milea, M., Frasincar, F. & Kaymak, U. (2009) tOWL: Integrating Time in OWL. In: Semantic Web Information Management, pp. 225-246. Pauwels, P. & Terkaj, W. (2019) ifcOWL ontology (IFC4x1). In: <https://standards.buildingsmart.org/IFC/DEV/IFC4/>, Accessed date: 25th January 2023. Pauwels, P., Zhang, S. & Lee, Y.C. (2017) Semantic web technologies in AEC industry: a literature overview. In: Automation in Construction 73 (2017), pp. 145-165 Rasmussen, M. H., Lefrançois, M., Schneider, G. & Pauwels, P. (2020) BOT: the Building Topology Ontology of the W3C Linked Building Data Group. In: Semantic Web. DOI: 10.3233/SW-200385. Rasmussen, M. H., Lefrançois, M., Bonduel, M., Hviid, C.A. & Karlshøj, J. (2018) OPM: An ontology for describing properties that evolve over time. In: 6th Linked Data in Architect and Construction Workshop. London, United Kingdom. Tanaka, F., Tsuchida, M., Onosato, M., Date, H., Kanai, S., Hada, Y., Nakao, M., Kobayashi, H., Hasegawa, E., Sugawara, T. & Oyama, T. (2018) Bridge Information Modeling based on IFC for supporting maintenance management of existing bridges. In: 17th International Conference on Computing in Civil and Building Engineering (ICCCBE) 2018. Tampere, Finland. Werbrouck, J., Pauwels, P., Beetz, J. & Mannens, E., (2022) Lbdserver-a federated ecosystem for heterogeneous linked building data. Semantic Web Journal (submitting). Ye, X., &König, M., (2021) Framework for automated billing in the construction industry using BIM and smart contracts. In Proceedings of the 18th International Conference on Computing in Civil and Building Engineering: ICCCBE 2020 (pp. 824-838). Springer International Publishing. Zheng, Y., Törmä, S. & Seppänen, O. (2021) A shared ontology suite for digital construction workflow. In: Automation in Construction 132 (2021), pp. 103930

## EC3 OCCP 2025

Paper zur OCCP (unveröffentlicht, aber eingereicht... ;-)), hier leider ohne die Abbildungen (wie auch in dem paper vorher):Abstract The Ontology for Chronological Construction Processes (OCCP) provides a semantic foundation for a chronological structure aimed at managing information in the Architecture, Engineering, Construction and Operations (AECO) sector. Based on the W3C TIME ontology (Cox et al., 2006), the OCCP introduces a lifecycle-oriented concept tailored to Building Information Modelling (BIM) workflows and advances the accurate management of time stamps by introducing phases, cycles, milestones and transitions to ensure traceability of states throughout and beyond the lifetime of assets. This paper describes the concept and composition of OCCP, its SHACL-based validation rules, and provides an example to demonstrate its practical application.Introduction The AECO sector faces growing challenges in managing complex project information effectively. While Building Information Modeling (BIM) provides a structured approach, current workflows lack robust mechanisms for integrating and utilizing time-sensitive data. Existing standards like Industry Foundation Classes (IFC) (ISO 16739-1) and Information Container for Linked Document Delivery (ICDD) (ISO 21597-1) ensure spatial and semantic interoperability, but their support for temporal data remains fragmented. This results in disconnected records of planned and actual timelines, leading to errors, inefficiencies, and reduced traceability (Autodesk, 2018). Current approaches, such as XML and relational databases, provide only basic timestamp storage, failing to capture semantic relationships between lifecycle stages. They also struggle with scalability, consistency, and integration within BIM and multimodal frameworks, issues that graph-based ontologies address more effectively (Berners-Lee et al., 2001). The W3C TIME ontology provides a foundational model for representing time-related information, but lacks domain-specific constructs needed for chronological consistency, overlapping phases, iterative cycles, and project-specific adaptations. The Ontology for Chronological Construction Processes (OCCP) introduces a structured approach for managing construction lifecycles by defining phases, cycles, transitions, and instants, with SHACL rules ensuring logical sequencing and data integrity. Designed for seamless integration with BIM and ICDD, OCCP enhances temporal data management, lifecycle validation, and modeling flexibility. This paper presents its core structure, logic, and applications, demonstrating its potential to transform time-data management in the AECO sector.State of the art The AECO sector increasingly relies on digital tools and standards to manage complex building projects. IFC, developed by buildingSMART International, has become the de facto standard for BIM interoperability, supporting geometric, semantic, and topological data exchange across disciplines (Sacks et al., 2018; buildingSMART International, 2023). While IFC provides detailed lifecycle management, it lacks robust support for dynamic temporal and process-oriented data (Pauwels et al., 2017). Multimodels emerged to address the limitations of single, centralized models (Scherer & Schapke, 2011). These integrate multiple specialized data sources (e.g., cost, schedule, energy models) into a unified project representation (Borrmann et al., 2018; Grilo et al., 2010). By linking different data layers, multimodal containers enhance information accessibility (Xie et al., 2024), but their lack of semantic integration and validation mechanisms can lead to inconsistencies and reduced reliability (Singh et al., 2011). Similarly, while ICDD (ISO 21597-1) improves traceability and version control, it does not inherently support semantic reasoning or structured chronological data management (Pauwels et al., 2016). Ontologies, expressed using RDF (Lassila & Swick, 1999), provide a powerful approach for structuring, validating, and reasoning about BIM data (Beetz et al., 2009). They improve data consistency and interoperability, making them suitable for semantic integration. The W3C TIME Ontology (Cox et al., 2006) is widely used for temporal data representation, supporting instants, intervals, and relationships such as time:Before and time:After. It enables integration with IfcOWL and ICDD for semantic temporal modeling (Pauwels et al., 2016) and facilitates advanced temporal queries, such as detecting overlapping project phases (Car et al., 2024). However, TIME lacks domain-specific constructs for AECO workflows, such as phases, iterative cycles, and planned vs. actual timestamps (Volk et al., 2014). Furthermore, it does not inherently validate chronological consistency, making it difficult to detect conflicts like overlapping timelines (Karlapudi et al., 2021). To address these shortcomings, SHACL enables the validation of RDF-based temporal structures (Knublauch et al., 2017; Debruyne et al., 2020), allowing the enforcement of logical constraints in AECO workflows. However, current IFC implementations still lack structured temporal representations, leading to fragmented data across multiple tools (Volk et al., 2014; Pauwels, 2017). Similarly, ICDD and multimodal containers structure heterogeneous data but lack inherent mechanisms for ensuring temporal consistency (Singh et al., 2011). Versioning systems, such as Global Information Tracker (GIT), Apache Subversion (SVN), and Ontology-based Programming Models (OPM), are widely used for tracking changes in software development and ontologies. However, these methods are not inherently designed for structured chronological modeling in AECO workflows, where updates must be linked to specific lifecycle events. The OCCP addresses this gap by integrating chronological records directly within the semantic model. Through the use of IFC-linked timestamps, OCCP captures the evolution of a component across different versions, providing a bidirectional version tracking system that can be used in combination with GIT-based repositories. This enables a fine-grained traceability approach, allowing stakeholders to track changes within both the IFC model and its associated temporal records, ensuring complete transparency in model evolution.The core concept of OCCP The OCCP is based on the W3C TIME ontology and extends its classes and object properties to provide a semantic representation of the building lifecycle. This ontology is part of the concept of chronological models (cMod) (Vaatz et al., 2023), which combines IFC-based models with the OCCP and other ontologies, such as the Bridge Topology Ontology (BROT) (Hamdan et al., 2020) or the Building Topology Ontology (BOT) (Rasmussen et al., 2020), and model-related data sets using ICDD. In this concept, the OCCP provides the temporal structure for storing the timestamps of model-related data and assigning the correct phase or milestone within the lifecycle. The general temporal structure of the OCCP, as shown in Figure 1, is designed to ensure applicability to a wide range of AECO projects by not adapting any national specific construction process definitions such as the German HOAI (Honorarordnung für Architekten und Ingenieure).Methodology of development The development of the OCCP followed a structured methodology based on domain-specific requirements, expert insights, and semantic validation. Given the complexity of AECO workflows, the ontology was designed to represent real-world temporal constructs, lifecycle transitions, and iterative review cycles while ensuring adaptability across different project structures by extending the W3C TIME ontology with specialized classes and properties, enabling dependency tracking, logical sequencing, and versioned event recording. Industry expert input further refined OCCP’s classifications, ensuring it accommodates iterative planning, review processes, and real-world deviations such as delays, rework cycles, or phased repairs. OCCP’s structure and reasoning capabilities were validated using a competency question-driven approach, where key queries tested the ontology’s ability to structure, retrieve, and enforce chronological consistency. These competency questions were formalized into SHACL rules, ensuring correct sequencing of phases, review cycles before approvals, and structured tracking of IFC updates. Among others, the following three competency questions guided the validation of OCCP’s semantic constraints: CQ1: How many review cycles occurred before approval, and how did this impact the planned vs. actual completion time? CQ2: What changes were made to an IFC component during planning and review? CQ3: When was an event recorded, and how many updates were made? These questions demonstrate the information demand that OCCP is designed to answer and OCCP’s ability to track changes, compare planned vs. actual schedules, and maintain structured, machine-readable chronological records within BIM and multimodal environments. Classes The W3C TIME ontology defines temporal entities, distinguishing between time:Instant (specific points in time without duration) and time:Interval (spans between two instants). The OCCP extends the TIME ontology by introducing occp:Phase and occp:Cycle as subclasses of time:Interval. All phases shown in Figure 1 (e.g., the occp:PhaseE\_Usage) are subclasses of occp:Phase, while cycles, such as occp:CycleA\_PlanningReview, capture iterative processes like multiple review cycles in planning. To structure milestone events, OCCP extends time:Instant by defining occp:Transition and phase-related instants. Transitions mark key milestones between phases or trigger subsequent project steps, such as occp:Submission, which signals the transition from planning to construction. Similarly, occp:CompletionOfPlanning defines the finalization of planning, ensuring proper sequencing of activities. Phase-related instants, such as occp:EditBSTR (BSTR - before submission to review) or occp:BeginningOfPlanning, provide precise temporal markers within each phase. Following this pattern, OCCP introduces dedicated event classes for each defined phase. Figure 2 illustrates the OCCP class hierarchy, showcasing its extension of the TIME ontology, while Figure 3 provides a detailed representation of the temporal framework for the first three phases, their transitions, and associated instants. Figure 3 highlights OCCP’s structured approach to temporal modeling, demonstrating how occp:PhaseA\_Instant, occp:PhaseB\_Instant, and occp:PhaseC\_Instant organize phase-specific instants within project lifecycles.Figure 2: General class hierarchy of OCCP The occp:BeginningOfPlanning serves as a foundational event marking the start of planning, while occp:SubmissionToReview signifies the handoff between planning and review. Transitions such as occp:Submission and occp:CompletionOfConstruction define clear boundaries between phases, supporting chronological consistency and milestone validation. The subclass relationships in Figure 3 illustrate how OCCP seamlessly integrates into the TIME ontology while introducing domain-specific instants and transitions. The ontology’s modular structure enhances semantic clarity, distinguishing between different temporal entities through explicit object properties. This modularity ensures adaptability to diverse project needs, supporting both high-level lifecycle planning and granular activity tracking.Figure 3: Extract of phase-related instant classes of OCCP Object properties The TIME ontology provides fundamental object properties for describing general relationships between time:Instant and time:Interval, but lacks the expressiveness needed for structured lifecycle management in AECO projects. The OCCP extends these capabilities by introducing additional object properties, as shown in Figure 4, refining both the owl:topObjectProperty and time:hasTime (including time:hasBeginning and time:hasEnd) to define chronological relationships between phases, cycles, milestones, and instances. To establish clear temporal associations, OCCP introduces occp:hasPhase and occp:isInPhase. The occp:hasPhase property embeds events within structured lifecycle phases, ensuring temporal instances are correctly classified. For example, occp:SubmissionToReview, which represents the submission of a completed plan for review, is explicitly linked to Phase A (Planning), maintaining a logically ordered structure for querying and validation. This ensures event traceability and enables automated reasoning over project lifecycles. The occp:isInPhase property, in contrast, captures nested or overlapping processes within phases. A practical example is the repair of a damaged component during the Usage phase, where the repair lifecycle must be contextualized within the broader phase. The occp:isInPhase relationship enables the representation of concurrent activities, acknowledging the reality of complex, interwoven construction processes. Beyond these structural properties, OCCP also refines temporal granularity by distinguishing between actual and estimated time values. The occp:hasActualTime and occp:hasEstimatedTime properties differentiate between verified timestamps and planned or uncertain events. Similarly, occp:hasActualBeginning and occp:hasEstimatedEnd extend time:hasBeginning and time:hasEnd, allowing precise modeling of both anticipated and confirmed project timelines. To enforce logical sequencing, OCCP introduces relational temporal properties such as occp:beginsBefore, occp:beginsAfter, occp:beginsWith, occp:endsBefore, occp:endsAfter, and occp:endsWith, enabling the semantic representation of phase dependencies and overlaps. Additionally, hierarchical lifecycle properties (occp:startsCycle, occp:endsCycle, occp:startsPhase, and occp:endsPhase) structure iterative processes, supporting the modular definition of project milestones, transitions, and recurrent events. This enriched semantic framework ensures that OCCP not only tracks time-related data, but also structures, validates, and integrates it within complex lifecycle models, supporting flexibility, consistency, and traceability in construction project management.SHACL ruleset The SHACL ruleset embedded in OCCP plays a critical role in validating and ensuring the logical consistency of temporal and semantic data. By defining and enforcing structural and semantic constraints on RDF data, SHACL ensures that relationships and properties within OCCP adhere to predefined logical rules. This validation layer bridges the gap between theoretical modeling and practical implementation, enabling robust verification of temporal and semantic correctness in an OCCP-enhanced framework. A primary function of the SHACL ruleset is to maintain the integrity of OCCP’s temporal structure by governing relationships between phases, transitions, and instants. Rules enforce sequential and hierarchical order, ensuring that each phase has a well-defined beginning and end, validated using properties such as occp:hasActualBeginning and occp:hasActualEnd. This mechanism prevents erroneous overlaps or gaps between phases—an essential requirement in multi-stakeholder construction projects, where precise chronology is crucial for lifecycle management. A distinct feature of OCCP’s SHACL ruleset is its ability to validate iterative and nested temporal structures. Within the planning phase, for instance, iterative cycles such as occp:CycleA\_PlanningReview regulate review processes. SHACL shapes ensure that each iteration only begins upon a defined trigger event, such as occp:ReviewRejection, and concludes with either another iteration or occp:ReviewApproval. This ensures logical sequencing and enhances traceability by embedding semantic rules directly into the data model. Beyond internal consistency, SHACL supports cross-contextual validation, enabling OCCP to accurately represent nested lifecycles and parallel workflows. For example, when a construction component undergoes a local repair within an ongoing broader construction phase, SHACL rules validate that the repair lifecycle—including its planning, execution, and completion phases—aligns with the overarching temporal constraints of the primary phase. In addition, SHACL enforces temporal constraints between milestones and transitions. Rules tied to properties such as occp:startsPhase and occp:endsPhase ensure that key events occur in correct sequence, e.g., occp:CompletionOfPlanning must precede the start of subsequent phases. Furthermore, these rules verify that estimated timestamps (occp:hasEstimatedTime) and actual timestamps (occp:hasActualTime) maintain logical consistency, preventing misalignment between planned and actual project timelines. A key strength of OCCP’s SHACL framework is its modularity, which enables project-specific customization without compromising the core ontology structure. Users can extend OCCP’s validation mechanisms to comply with regional planning standards, industry-specific constraints, or unique project milestones. This adaptability ensures OCCP’s applicability across diverse AECO workflows. In practice, SHACL enhances data validation workflows by providing explicit violation reports when data fails to meet predefined constraints. If a phase violates temporal boundaries or a transition lacks proper phase alignment, SHACL generates clear diagnostic feedback. This automated validation is particularly valuable in dynamic BIM and multi-model environments, where continuous updates necessitate rigorous and consistent verification to maintain data accuracy and reliability. Examples for OCCP application Figure 5 presents a multi-layered example demonstrating how OCCP structures time records and links them to phase-related instances. Layer 1 illustrates the sequential order of phases (Block 1), while Block 2 shows how a component individual (semantic representation) connects to phase-related instances via occp:hasInstant, linking components to occp:Phase(X)\_Instants. Each instant is assigned a timestamp (e.g., via time:hasTime) as indicated in Legend 2. The main diagram depicts six blue events within the planning phase and four red events within the review phase. Event 1 (Ind. 1) marks the beginning of planning, which automatically initiates a new planning phase (IND:PhaseA\_Planning) and a new planning-review cycle (IND:CycleA\_PlanningReview). After the first submission for review (Ind. 4), the review phase starts (IND:PhaseB\_Review) through an OCCP SHACL rule, which ensures that occp:startsPhase properly initiates the review phase. A rejection (Ind. 6) ends the current cycle (Ind. I.), prompting a new iteration (Ind. II.) with IND:PhA\_Edit\_ASTR (edit after submission to review). Once the review is approved (Ind. 7), the 2nd cycle (Ind. II.) and phases A and B (Ind. A & B) conclude, marking the transition to construction (Ind. C). This example demonstrates how OCCP records and connects temporal events across phases, ensuring that their relationships and sequential logic are semantically structured. The same approach applies to subsequent phases, such as construction (Phase C) or usage (Phase D), as well as transitions like submission or completion of construction. Example 2 extends this logic by embedding the planning-review cycle from Figure 5 into a repair phase using occp:isInPhase, creating a nested lifecycle. This scenario, applicable to cases such as structural damage repair, could represent a bridge bearing replacement, where the damage is recorded through inspections and reaches a critical threshold, requiring immediate intervention. The repair lifecycle, containing its own planning, review, and execution phases, coexists with and impacts the broader usage phase (occp:PhaseD\_Usage), affecting the overall bridge operation. OCCP enables clear documentation of these interdependencies, ensuring full traceability of how the repair process affects and integrates with the broader lifecycle of the structure. Figure 6 illustrates how OCCP handles estimated vs. actual timestamps. At the start of the review phase (Ind. 5), an estimated completion date is set. However, a rejection event (Ind. 6) extends the timeline, requiring additional planning iterations until final approval. The actual end date is only assigned once the reviewer grants approval, providing a clear distinction between planned and actual progress in project execution. These examples highlight OCCP’s capability to structure, validate, and adapt lifecycle representations across varied use cases. The ontology’s SHACL ruleset enforces compliance with defined chronological relationships, ensuring logical consistency while allowing flexibility for diverse project requirements. Answering competency questions with OCCP The example shown in Figures 5 and 6 illustrates OCCP's ability to structure lifecycle events and track updates to IFC components within planning, review, and repair cycles. The following competency questions further demonstrate how OCCP provides structured answers to key project-related queries: 1. How many review cycles occurred before approval, and how did this impact the planned vs. actual completion time? • OCCP records each iteration of a review cycle using occp:CycleA\_PlanningReview, which can be counted to determine the number of review attempts before approval. • The estimated and actual phase completion times, stored under occp:hasEstimatedEnd and occp:hasActualEnd, allow for direct comparisons of planned vs. actual timelines. 2. What changes were made to an IFC component during planning and review? • Updates to IFC elements are recorded as instances of occp:Update, linking components to modification timestamps (occp:hasUpdateTime) and property changes (occp:hasUpdatedProperty). • The ontology maintains a record of previous (occp:hasPreviousValue) and new (occp:hasNewValue) values for each change, ensuring full traceability. 3. When was an event recorded, and how many updates were made? • The first recording of an event is tracked using occp:hasRecordingTime, while subsequent modifications are logged with occp:hasUpdateTime, allowing for a reconstruction of event history. Figure 7 provides a compact RDF/Turtle example illustrating how OCCP can structure this data. To validate this information, the following SHACL constraint as shown in Figure 8 ensures that all updates to IFC components include a valid occp:hasUpdateTime and Figure 7: Example of data structuring in OCCPreference an updated property. Due to space limitations, only the core validation rules concerning the modification of recorded events, IFC components, and temporal properties are shown. The complete ruleset encompasses additional constraints to ensure chronological consistency across various lifecycle events, covering scenarios such as sequential update dependencies, version tracking across multiple IFC identifiers, and logical validation of estimated versus actual timestamps. These rules collectively reinforce OCCP’s ability to maintain structured, machine-readable chronological records while preserving data integrity. By enforcing conditions on updates through SHACL, OCCP ensures that changes in recorded lifecycle information remain transparent, verifiable, and systematically integrated within multimodal BIM workflows.Figure 8: Minimal example of SHACL rules for updates Outlook and conclusion The OCCP is nearing public release, including its source code and documentation, marking a significant milestone in its development. The next phase focuses on its practical implementation within chronological models (cMod), designed to unify lifecycle-based temporal data with broader lifecycle information management (LIM) systems. The cMod framework will operationalize OCCP’s semantic structure, enabling real-time validation, version tracking, and predictive analysis. By linking model-specific OCCP (MsOCCP) datasets to IFC models and supporting graph-based reasoning, cMod will facilitate automated consistency checks and lifecycle simulations. A dedicated software interface is under development to support cMod creation, management, and model-stage visualization. OCCP extends the W3C TIME ontology with domain-specific constructs such as phases, cycles, transitions, and phase-specific instants, providing a structured framework for integrating time-based data into BIM workflows. Its SHACL-based validation ensures logical consistency, minimizing errors and improving data reliability. By supporting iterative cycles, overlapping phases, and complex lifecycle structures, OCCP enhances timeline visibility, traceability, and automated consistency checks. The introduced update tracking mechanism strengthens OCCP’s ability to record, query, and validate IFC modifications, ensuring seamless version tracking and bidirectional synchronization with Git-based repositories. Furthermore, OCCP was tested against domain-relevant competency questions, demonstrating its capability to retrieve key lifecycle insights, validate planned vs. actual schedules, and track iterative processes such as review cycles and component modifications. The competency-question-driven approach, supported by SHACL validation, ensures that OCCP does not merely store time-related data but actively verifies chronological consistency and logical sequencing. As a flexible and extensible ontology, OCCP contributes to advancing lifecycle-oriented construction informatics. While further refinement is needed to optimize its integration with industry tools, its structured approach to time management in BIM and multimodal workflows provides a solid foundation for improving consistency, traceability, and automation of chronological records. Future work will focus on expanding its application, enhancing interoperability, and evaluating its real-world impact through practical case studies and software implementations. References Autodesk (2018) Construction Disconnected: The High Cost of Poor Data and Miscommunication. <https://www.autodesk.com/blogs/construction/construction-disconnected-fmi-report/> . Accessed date: 30th January 2025. Beetz, J., van Leeuwen, J. P., & de Vries, B. (2009) IfcOWL: A case of transforming EXPRESS schemas into ontologies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 23(1), 89-101. DOI: 10.1017/S0890060409000122 Berners-Lee, T., Hendler, J. & Lassila, O. (2001) The Semantic Web. Scientific American, 284(5), 34-43. <https://doi.org/10.1038/scientificamerican0501-34> Borrmann, A., König, M., Koch, C., & Beetz, J. (2018) Building Information Modeling: Technology Foundations and Industry Practice. Springer International Publishing. DOI: 10.1007/978-3-319-92862-3 buildingSMART International (2023) Industry Foundation Classes (IFC) – Standard. <https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/> . Accessed date: 30th January 2025. Car, N.J., Homburg, T., Perry, M., Knibbe, F., Cox, S.J.D., Abhayaratna, J., Bonduel, M., Cripps, P.J., & Janowicz, K. (2024). OGC GeoSPARQL - A Geographic Query Language for RDF Data (Version 1.1). Open Geospatial Consortium (OGC). <https://docs.ogc.org/is/22-047r1/22-047r1.html>. Accessed date: 29th January 2025. Cox, S. & Little, C. (2006) Time Ontology in OWL. <https://www.w3.org/TR/owl-time/> . Accessed date: 29th January 2025. Debruyne, C. & McGlinn, K. (2020). Reusable SHACL Constraint Components for Validating Geospatial Linked Data. In: GeoLD@ESWC 2020 – Proceedings of the 1st International Workshop on Geospatial Linked Data. Heraklion, Greece. <https://chrdebru.github.io/papers/2021-geold-preprint.pdf> . Accessed date: 30th January 2025. Fuchs, S., Kaddolsky, M. & Scherer, R. J. (2011) Formal description of a generic multi-model. In 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 205-210. Grilo, A., & Jardim-Goncalves, R. (2010) Value proposition on interoperability of BIM and collaborative working environments. Automation in Construction, 19(5), 522-530. DOI: 10.1016/j.autcon.2009.11.003 Hamdan, A. H. & Scherer, R. J. (2020) Integration of BIM-related bridge information in an ontological knowledgebase. In: Proceedings of the 8th Linked Data in Architecture and Construction Workshop (LDAC) Karlapudi, J., Valluru, P., & Menzel, K. (2021). Ontology Approach for Building Lifecycle Data Management. Proceedings of the ASCE International Conference on Computing in Civil Engineering (i3CE2021). Orlando, Florida, USA. Knublauch, H. & Kontokostas, D. (2017). Shapes Constraint Language (SHACL). <https://www.w3.org/TR/shacl/> . Accessed date: 30th January 2025. Lassila, O., & Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. <https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/> . Access date: 30th January 2025. Pauwels, P., Zhang, S., & Lee, Y.-C. (2017). Semantic web technologies in AEC industry: A literature overview. Automation in Construction, 73, 145–165. <https://doi.org/10.1016/j.autcon.2016.10.003> Pauwels, P., & Terkaj, W. (2016) EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction, 63, 100-133. DOI: 10.1016/j.autcon.2015.12.003 Rasmussen, M. H., Lefrançois, M., Schneider, G. & Pauwels, P. (2020) BOT: the Building Topology Ontology of the W3C Linked Building Data Group. In: Semantic Web. DOI: 10.3233/SW-200385. Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). BIM Handbook: A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers (3rd ed.). John Wiley & Sons. DOI: 10.1002/9781119287568 Scherer, R.J., & Schapke, S.-E. (2011). A distributed multi-model-based management information system for simulation and decision-making on construction projects. Advanced Engineering Informatics, 25(4), 582–599. Singh, V., Gu, N. & Wang, X. (2011) A Theoretical Framework of a BIM-Based Multi-Disciplinary Collaboration Platform. Automation in Construction, 20(2), 134-144. DOI: 10.1016/j.autcon.2010.09.011 Vaatz, A., Hamdan, A.-H., Al-Sadoon, N., Wogan, M., Menzel, K., (2023). Integration of semantic temporal information in BIM using ontologies. European Conference on Computing in Construction. Crete, Greece, July 10-12. DOI: 10.35490/EC3.2023.281 Volk, R., Stengel, J., & Schultmann, F. (2014). Building Information Modeling (BIM) for existing buildings — Literature review and future needs. Automation in Construction, 38, 109–127. <https://doi.org/10.1016/j.autcon.2013.10.023> Xie, Y., Zhan, N., Zhu, Q., Zhan, J., Guo, Z., Qiao, C., Zhu, J., & Xu, B. (2024). Multimodal data visualization method for digital twin campus construction. International Journal of Digital Earth, 17(1), 145–165. <https://doi.org/10.1080/17538947.2024.2431624>

EG ICE cMod 2025 – extended Abstract

Hier noch das (ebenfalls unveröffentlichte, aber eingereichte und akzeptierte (!!! ;-)) extended abstract zum cMod: Chronological Models (cMod) Towards consistent lifecycle information managementKEYWORDS Lifecycle information management, OCCP, model chronology, ontology 1. INTRODUCTION Effective temporal data management is a pressing challenge in construction projects, where accurate tracking of progress, changes and state transitions is essential. While BIM provides a strong foundation for data interoperability, it lacks tools to manage chronological information with precision. Filling this gap, Chronological Models (cMod) (Vaatz, A., et al., 2023) enable detailed time management by integrating timestamps, phase-specific data and lifecycle tracking to enable a Total Life Cycle Process (TLCP) (Wolf, G., et al., 2024) directly into BIM workflows. The cMod Manager (cMM) acts as a central tool that connects IFC models to semantic temporal frameworks. It enables real-time validation, advanced querying and granular documentation of construction progress. This paper explores the concept of cMod and shows how cMM operationalises temporal management to increase transparency and efficiency in construction workflows. 2. CORE CONCEPT OF CMOD Chronological Models aim to bridge the gap between static BIM data and the dynamic temporal needs of lifecycle information management (LIM). At its core, cMod provides a framework for embedding temporal information directly into construction models, enabling changes and progress to be tracked throughout the project lifecycle. Central to cMod is the ability to store, validate and visualise temporal data such as timestamps, phase transitions and historical states of components. The cMod approach is based on three key pillars: temporal enrichment of IFC models, integration with semantic frameworks, and advanced data validation. By enriching IFC models with time stamps and linking them to semantic data structures, cMod ensures chronological consistency while maintaining compatibility with industry standards and concepts like applying ontology-based rules for infrastructure planning (Wogan, M., et al., 2024), (Mellenthin Filardo, M., et al., 2024) or other ontologies, such as the Damage Topology Ontology (DOT) (Hamdan, A.-H., et al., 2021). It supports a variety of use cases, including progress monitoring, phase-specific comparisons and detailed audits of planned versus actual timelines. 2.1 Main elements The cMod concept revolves around three key modules that together provide a comprehensive framework for managing and validating time data in construction projects: 1. OCCP: The Ontology for Chronological Construction Processes (OCCP) serves as the semantic foundation for the lifecycle-based chronology. It provides the underlying structure for defining phases, transitions, cycles and events, allowing a flexible and ontology-driven approach to temporal data management. OCCP ensures logical consistency and a structured representation of the timeline, which is the basis for all cMod-related processes. 2. Model specific OCCP (MsOCCP): The MsOCCP is an IFC model-specific dataset in Turtle format that semantically structures temporal information. By enriching IFC components with temporal properties defined by OCCP, MsOCCP captures the chronology of building components. This dataset enables advanced queries, temporal validation and seamless integration of planned and actual data, bridging the gap between static IFC models and dynamic project schedules. 3. cMod Manager (cMM): The cMod Manager is a dedicated software application for creating, managing and validating cMods. It acts as an interface between IFC models, MsOCCP datasets and the user, providing tools to visualise, edit and ensure the consistency of temporal data. The cMM streamlines the handling of complex construction timelines, making them accessible and manageable within a user-friendly framework. 2.2 Components of the cMod Manager The cMM is the operational core of the cMod framework and facilitates the integration, management and validation of temporal data. Its functionality is centred on several main components that ensure seamless data processing and validation: 1. IFC integration: The cMM imports IFC models to extract structural and property information. These models serve as the basis for associating temporal data, allowing the enrichment of structural elements with MsOCCP properties. 2. OCCP-based validation: Using OCCP as the semantic backbone, the cMM validates MsOCCP records against SHACL-defined rules. This ensures logical consistency within the temporal structure and verifies compliance with lifecycle definitions. 3. MsOCCP Management: The cMM provides tools for creating, editing and visualising MsOCCP records. It allows users to add new timestamps, define transitions and validate changes in real time. 4. Hash-based change tracking: To maintain data integrity, the cMM includes a hash-based change tracking mechanism for IFC models. This approach detects changes, additions or deletions to components, ensuring a consistent link between the IFC model and its associated MsOCCP data. 5. Git integration: The cMM integrates with Git for advanced version control. This allows users to maintain a detailed history of changes, branch management and collaboration. Every change in the cMM is committed to a Git repository, ensuring traceability and the ability to revert to previous states. 6. Information flow: The cMM facilitates a streamlined data exchange process. IFC models are loaded and linked to MsOCCP records, validated against OCCP rules and enriched with time information. Any changes are versioned using Git, with hash values ensuring consistency across all data layers. By combining semantic reasoning, validation mechanisms and robust versioning, the cMM provides a practical solution for managing complex chronological, model-related data in construction workflows. 7. CONCLUSION Chronological Models offer a transformative approach to time management in construction projects, addressing critical limitations in traditional BIM workflows. Combining semantic data integration, real-time validation and advanced visualisation, cMod enables accurate tracking of construction progress and status. The cMod Manager serves as the cornerstone, operationalising the concept and facilitating its integration into practical workflows. This paper highlights the potential of cMod to improve transparency, efficiency, data consistency and accountability in construction management, and lays the foundation for future advances in temporal data integration and building information management. REFERENCES Hamdan, A.-H., Taraben, J., Helmrich, M., Mansperger, T., Morgenthal, G. & Scherer, R. J., (2021). A semantic modeling approach for the automated detection and interpretation of structural damage. Automation in Construction, 128, p.103739. DOI: <https://doi.org/10.1016.j.autcon.2021.103739>. Mellenthin Filardo, M. & Liu, Li. & Hagedorn, P., Zentgraf, S., Melzner, J. & König, M., (2024). A standard-based ontology network for information requirements in digital construction projects. Linked Data in Architecture and Construction. Bochum, GER, June 13-14. Available: <https://ceur-ws.org/Vol-3824/paper6.pdf> Vaatz, A., Hamdan, A.-H., Al-Sadoon, N., Wogan, M., Menzel, K., (2023). Integration of semantic temporal information in BIM using ontologies. European Conference on Computing in Construction. Crete, Greece, July 10-12. DOI: 10.35490/EC3.2023.281 Wogan, M., Meyer, N., Mohan, N., Schilling, S., Vaatz, A., Lelke, T., Menzel, K., Gross, R., (2024). Digital Rules in Infrastructure Planning: Presentation of an Ontology-Based Approach, p. 31-40. EG-ICE 2024. Vigo, Spain, July 03-05. Available: <https://3dgeoinfoeg-ice.webs.uvigo.es/proceedings> Wolf, G., Wogan, M., Mohan, N., Nejat, N., Gross, R., Menzel, K., (2024). Advancing AEC Project Management: A Model-Based and Data-Driven Approach for Sustainable Practices, p. 784-789. European Conference on Computing in Construction. Chania, Greece, July 15-17.DOI: <http://www.doi.org/10.35490/EC3.2024.198>

## cMod-Konzept

Das ist alles, was ich bisher im wissenschaftlichen Kontext dazu geschrieben habe. Der Rest ist einerseits konzeptionelle Arbeit (die ich größtenteils mit ChatGPT diskutiert habe - btw: Mein bisheriger Eindruck Dir ist, dass Du sehr viel schneller, präziser, aktueller und fokussierter bist und ich freue mich sehr auf die Weiterführung der Entwicklung mit Deiner Unterstützung!) und programmiertechnische Arbeit, wobei mir auch ChatGPT geholfen hat. Dabei bin ich jedoch recht schnell an die Grenzen von ChatGPT gestoßen, deshalb will ich nun schauen, ob Du mir da weiterhelfen kannst. (Mein letztes Problem war die Implementierung des Pellet-Reasoners in mein Validierungs-Modul, das habe ich mit ChatGPT innerhalb von ca. 4 Tagen nicht hinbekommen, mit Deiner Unterstützung war es in einer Stunde erledigt - und das liegt vor allem daran, dass ich ziemlich schwer von Begriff bin!! xD Aber das wirst Du sicherlich noch merken... ;-) :-D) Ich starte mal mit dem Konzept, wobei ich betonen muss, dass das alles nicht in Stein gemeißelt ist und nur eine Möglichkeit der Umsetzung darstellt. Ich möchte möglichst nicht zu viel vorgeben hinsichtlich der Umsetzung, denn ich möchte auch mit Deiner Hilfe herausfinden, ob meine Ansätze überhaupt sinnvoll, zweckerfüllend und umsetzbar sind. Ich entschuldige mich für die Form und die störenden Absätze, es handelt sich bei dem folgenden Teil um einen aus ChatGPT herauskopierte Antwort, die das Ergebnis einer sehr langen Unterhaltung war. Here we go:1. Grundlegender Informationsfluss für ein cModSchritte im Informationsfluss: 1. Import des IFC-Modells: o Das IFC-Modell wird in die Software (cMod-Manager, kurz cMM) geladen. o Auslesen der Modellstruktur und der zugehörigen Eigenschaften. Hierfür wird ein IFC-Parser benötigt (z. B. IfcOpenShell für Python). 2. Auswahl einer Modellkomponente: o Die Benutzeroberfläche (UI) des cMM zeigt die Modellstruktur und ermöglicht die Auswahl einer Komponente (z. B. K1). o Auswahl über grafische Elemente (3D-Modell) oder Baumstruktur (listet alle Komponenten). 3. Analyse der Komponente: o Überprüfung, ob für die Komponente (K1) bereits Daten im IFC-Modell-spezifischen OCCP-Datensatz (MsOCCP) existieren. o Falls MsOCCP vorhanden ist: Validierung der bestehenden Daten anhand der Basis-OCCP-Regeln (über SHACL-Engine, z. B. PySHACL). 4. Darstellung und Bearbeitung der Daten: o Darstellung aller vorhandenen Einträge für K1, gruppiert nach Phase, Prozessschritt und Zeitstempel. o Möglichkeit zur Bearbeitung und Erstellung neuer Einträge (z. B. Hinzufügen eines neuen Zeitpunkts in einer Phase). 5. Speicherung der Änderungen: o Änderungen werden in den MsOCCP-Datensatz geschrieben. o Der aktualisierte MsOCCP wird im ICDD-Container gespeichert. 6. Export und Visualisierung: o Der Benutzer kann den Zustand des IFC-Modells zu einem beliebigen Zeitpunkt visualisieren. o Hierfür wird eine Abfrage gegen den MsOCCP durchgeführt, um relevante Zustände zu filtern und im IFC-Viewer anzuzeigen.

1. Datenformate und TechnologienDatenformate: • IFC (Industry Foundation Classes): Das Standardformat für Bauprojekte. o Speichert geometrische, topologische und semantische Informationen. • RDF/TTL (Turtle): Für MsOCCP, da es sich ideal für semantische Daten und Linked Data eignet. • ICDD (Information Container for Data Delivery): Containerformat für IFC, TTL und andere Dateien. Technologien: • IFC-Parsing: o IfcOpenShell: Ermöglicht das Laden, Bearbeiten und Speichern von IFC-Dateien. o BIMserver: Server-basierte IFC-Verwaltung. • RDF/OCCP-Datensatz: o SPARQL: Für Abfragen gegen den MsOCCP. o SHACL: Für Validierung der Daten. o Apache Jena Fuseki: RDF-Datenbank zur Speicherung und Abfrage des MsOCCP. • Verknüpfung von IFC und MsOCCP: o Linked Data mit RDF-Triple-Verknüpfungen zwischen IFC-Komponenten und OCCP-Daten. • UI/Viewer: o BlenderBIM: Visualisierung von IFC-Daten. o Custom 3D-Viewer (z. B. mit Three.js): Erlaubt gezielte Interaktion und Selektion.
2. Speicherung von ModellzuständenFrage: Ist es möglich, alle Zustände in einer IFC-Datei zu speichern?Die IFC-Datei ist nicht darauf ausgelegt, zahlreiche Versionen oder Zustände innerhalb einer Datei zu speichern. Das Speichern aller Modellzustände in einer einzigen IFC-Datei würde: • Unpraktikabel groß werden. • Die Lesbarkeit und Performance stark beeinträchtigen. Alternative Ansätze:
3. MsOCCP als Zustandsspeicher: o Speichere in der IFC-Datei nur die aktuelle Version. o Alle Zustände (Zeitpunkte, Phasen) werden im MsOCCP gespeichert. o Die IFC wird bei Abfragen dynamisch angepasst, um einen Zustand wiederzugeben.
4. Linked Data Ansatz: o Speichere Änderungen als separate RDF-Triple-Daten (z. B. „Eigenschaft X wurde an Zeit T geändert“). o Verknüpfe diese mit dem MsOCCP.
5. Delta-Speicherung: o Statt alle Zustände vollständig zu speichern, speichere nur Differenzen (Deltas) zwischen Zuständen.
6. IFC-Versionierung: o Falls alle Zustände gespeichert werden müssen: Nutze externe Tools wie Git für IFC-Dateien (mit IfcOpenShell-Diff).
7. Beispiel: InformationsflussDatenmodell
8. IFC-Modell: ifc Code kopieren # IFC-Komponente K1 # Beispiel einer Wand mit ID 1234 # Wandeigenschaften in der IFC-Datei gespeichert. ttl Code kopieren occp:K1 a occp:IFCComponent ; occp:hasPhase occp:PhaseA\_Planning ; occp:hasTime "2022-01-01"^^xsd:date .
9. MsOCCP (RDF-Daten): ttl Code kopieren occp:K1\_Milestone1 a occp:BeginningOfPlanning ; time:hasTime "2022-01-01"^^xsd:date ; time:before occp:K1\_Milestone2 . occp:K1\_Milestone2 a occp:DataProcurement ; time:hasTime "2022-01-02"^^xsd:date ; time:before occp:K1\_Milestone3 . Datenfluss
10. Der Benutzer lädt IFC und MsOCCP in die Software.
11. Die Software verbindet beide: IFCComponent K1 wird mit MsOCCP K1 verknüpft.
12. Der Benutzer bearbeitet Zeitstempel, z. B. fügt einen neuen Milestone hinzu.
13. Änderungen werden validiert und im MsOCCP gespeichert.
14. Ein neuer Container (ICDD) wird erzeugt. • Git als Versionierungstool: o Git speichert Änderungen (Deltas) zwischen verschiedenen Versionen des IFC-Modells. o Jeder Commit repräsentiert einen neuen Zustand des Modells (z. B. nach einer Bearbeitung). • Automatisierung der Git-Vorgänge: o Alle Git-Operationen (Commit, Branching, Merging) werden durch die Software (cMM) automatisiert. o Der Nutzer interagiert nur mit der Benutzeroberfläche, die Änderungen anzeigt und bearbeitet. • Verknüpfung mit OCCP: o OCCP-Zeitstempel und Phaseninformationen werden parallel in einem RDF-Datensatz (MsOCCP) gespeichert. o Änderungen im IFC-Modell werden mit OCCP-Informationen synchronisiert, sodass Git und MsOCCP immer konsistent bleiben. Umsetzungsschritte
15. Initialisierung: o Jedes Projekt (cMod) wird als Git-Repository angelegt. o IFC-Datei und MsOCCP-Datensatz werden als separate Dateien hinzugefügt.
16. Automatisierte Git-Operationen: o Commit: Jede Änderung am IFC-Modell oder MsOCCP wird automatisch versioniert: ♣ Benutzer bearbeitet eine Komponente im cMM. ♣ Änderungen werden gespeichert, und die Software erzeugt automatisch einen Commit. o Branching: Für größere Änderungen kann automatisch ein neuer Branch erstellt werden: ♣ Z. B. "Branch A" für die Planung, "Branch B" für die Überprüfung. o Merging: Wenn Änderungen aus verschiedenen Phasen kombiniert werden, führt die Software ein automatisches Merge durch, prüft jedoch zuerst die Konsistenz.
17. Visualisierung der Änderungen: o Die Benutzeroberfläche zeigt Änderungen in einer übersichtlichen Chronologie: ♣ cMM erstellt eine Historie aller Bearbeitungsschritte. ♣ Nutzer können eine ältere Version anzeigen und vergleichen.
18. Fehlerprüfung: o Vor jedem Commit prüft die Software: ♣ Ob das Modell gemäß OCCP konsistent ist. ♣ Ob Änderungen logisch sind (z. B. keine widersprüchlichen Zeitstempel).
19. Export und Bereitstellung: o Der aktuelle Zustand des Modells kann jederzeit exportiert werden. o Andere Nutzer können das Git-Repository direkt klonen oder über die Software eine Version laden.

Technologien und Tools • Git-Integration: o GitPython: Python-Bibliothek zur Steuerung von Git. o Automatische Commits und Merges. • IFC-Differenzierung: o IfcOpenShell-Diff: Zum Vergleichen von IFC-Dateien und Generieren von Deltas. • RDF-Datensatz: o Änderungen im MsOCCP werden mit Git synchronisiert. • UI/Frontend: o Anzeige und Bearbeitung der Modellhistorie (z. B. über eine Timeline).

Vorteile: 1. Konsistenz und Nachvollziehbarkeit: o Jeder Zustand des Modells ist gespeichert und nachvollziehbar. o OCCP-Zeitstempel garantieren logische Konsistenz. 2. Nutzerfreundlichkeit: o Git-Operationen werden vollständig durch die Software gesteuert. o Keine Git-Kenntnisse erforderlich. 3. Flexibilität: o Verschiedene Modellzustände (z. B. Planungs- und Prüfphasen) können separat entwickelt und später zusammengeführt werden. 4. Fehlererkennung: o Konflikte oder widersprüchliche Änderungen werden automatisch erkannt und gemeldet. Entwicklungsplan: [[cMM 1 Entwicklungsplan]]Ergänzungen und Merkliste: • Echtzeitprüfung bei der Dateneingabe: o Jede Änderung in der MsOCCP wird mit den Regeln in der OCCP.ttl abgeglichen. o SHACL-Engines wie pySHACL oder SPARQL-Engines können in die Software integriert werden, um diese Überprüfungen direkt auszuführen. • Benutzerfreundliche Fehleranzeige: o Wird eine Eingabe als fehlerhaft erkannt, sollte die Software die Ursache des Fehlers verständlich anzeigen und möglichst Korrekturvorschläge liefern. • Effizienz: o Die Validierung muss performant und möglichst ressourcenschonend sein, insbesondere bei größeren IFC-Modellen oder umfangreichen MsOCCP-Dateien. • Schrittweise Validierung: o Validierung auf Eingabenebene (beim Hinzufügen einzelner Instanzen). o Validierung der gesamten MsOCCP-Datei beim Speichern oder vor größeren Operationen.

Im Laufe der Entwicklung habe ich dann entschieden, dass ich alle Funktionen zur Verwaltung von Updates und Linked Data in eine separate Ontologie auslagere, die als modulare Erweiterung der OCCP verwendet werden kann. Diese beiden Ontologien möchte ich als logisches Fundament für die Software zum Management von cMods (cMM) nutzen. Zum Entwicklungsstand von cMM kommt später mehr.

# Ontologien – Entwicklungsstand

## OCCP - TBox

@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@base <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1> .

@prefix sh: <http://www.w3.org/ns/shacl#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix osh: <http://www.occpshape.de/shape#> .

<http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1> rdf:type owl:Ontology ;

owl:imports <http://www.w3.org/2006/time#> ;

rdfs:comment "First workaround to implement time tracking features to models"@en .

#################################################################

# Object Properties

#################################################################

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#after

occp:after rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf time:after ;

owl:inverseOf occp:before ;

rdf:type owl:TransitiveProperty ;

rdfs:comment "A transitive variant of time:after, ensuring that if A occurs after B and B occurs after C, then A also occurs after C."@en ;

rdfs:label "after (transitive)"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#before

occp:before rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf time:before ;

owl:inverseOf occp:after ;

rdf:type owl:TransitiveProperty ;

rdfs:comment "A transitive variant of time:before, ensuring that if A occurs before B and B occurs before C, then A also occurs before C."@en ;

rdfs:label "before (transitive)"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#beginsAfter

occp:beginsAfter rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasBeginning> ;

rdfs:domain <http://www.w3.org/2006/time#TemporalEntity> ;

rdfs:range <http://www.w3.org/2006/time#TemporalEntity> ;

owl:inverseOf occp:beginsBefore ;

rdf:type owl:TransitiveProperty ;

rdfs:comment "The object property \"beginsAfter\" is used with time:interval or time:instant to describe the chronological order of the beginnings of T1 and T2. If a time:interval T1 \"beginsAfter\" T2, it only describes, that the beginning of T1 happens after the beginning of T2 without defining the order of the ends of T1 and T2, so the ends of T1 and T2 can happen in any order or simulaneously."@en ;

rdfs:label "begins after"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#beginsBefore

occp:beginsBefore rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasBeginning> ;

rdfs:domain <http://www.w3.org/2006/time#TemporalEntity> ;

rdfs:range <http://www.w3.org/2006/time#TemporalEntity> ;

owl:inverseOf occp:beginsAfter ;

rdf:type owl:TransitiveProperty ;

rdfs:comment "The object property \"beginsBefore\" is used with time:interval or time:instant to describe the chronological order of the beginnings of T1 and T2. If a time:interval T1 \"beginsBefore\" T2, it only describes, that the beginning of T1 happens before the beginning of T2 without defining the order of the ends of T1 and T2, so the ends of T1 and T2 can happen in any order or simulaneously."@en ;

rdfs:label "begins before"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#beginsWith

occp:beginsWith rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasBeginning> ;

rdfs:comment "The object property \"beginsWith\" is used with time:interval or time:instant to describe the chronological order of the beginnings of T1 and T2. If a time:interval T1 \"beginsWith\" T2, it only describes, that the beginning of T1 happens on the same day with the beginning of T2 without defining the order of the ends of T1 and T2, so the ends of T1 and T2 can happen in any order or simulaneously."@en ;

rdfs:label "begins with"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#endsAfter

occp:endsAfter rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasEnd> ;

rdfs:domain <http://www.w3.org/2006/time#TemporalEntity> ;

rdfs:range <http://www.w3.org/2006/time#TemporalEntity> ;

owl:inverseOf occp:endsBefore ;

rdf:type owl:TransitiveProperty ;

rdfs:comment "The object property \"endsAfter\" is used with time:interval or time:instant to describe the chronological order of the endings of T1 and T2. If a time:interval T1 \"endsAfter\" T2, it only describes, that the ends of T1 happens after the end of T2 without defining the order of the beginnings of T1 and T2, so the beginnings of T1 and T2 can happen in any order or simulaneously."@en ;

rdfs:label "ends after"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#endsBefore

occp:endsBefore rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasEnd> ;

rdfs:domain <http://www.w3.org/2006/time#TemporalEntity> ;

rdfs:range <http://www.w3.org/2006/time#TemporalEntity> ;

owl:inverseOf occp:endsAfter ;

rdf:type owl:TransitiveProperty ;

rdfs:comment "The object property \"endsBefore\" is used with time:interval or time:instant to describe the chronological order of the endings of T1 and T2. If a time:interval T1 \"endsBefore\" T2, it only describes, that the end of T1 happens before the end of T2 without defining the order of the beginnings of T1 and T2, so the beginnings of T1 and T2 can happen in any order or simulaneously."@en ;

rdfs:label "ends before"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#endsCycle

occp:endsCycle rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasEnd> ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:range occp:Cycle ;

rdfs:comment "The object property \"endsCycle\" is used to mark instants that end an ongoing cycle within a phase."@en ;

rdfs:label "ends cycle"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#endsPhase

occp:endsPhase rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasEnd> ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "The object property \"endsPhase\" is used to mark instants that end an ongoing phase."@en ;

rdfs:label "ends phase"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#endsWith

occp:endsWith rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasEnd> ;

rdfs:comment "The object property \"endsWith\" is used with time:interval or time:instant to describe the chronological order of the endings of T1 and T2. If a time:interval T1 \"endsWith\" T2, it only describes, that the ends of T1 happens on the same day as (the end of) T2 without defining the order of the beginnings or points in time of T1 and T2, so the beginnings of T1 and T2 can happen in any order or simulaneously."@en ;

rdfs:label "ends with"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasActualBeginning

occp:hasActualBeginning rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasBeginning> ;

rdfs:domain <http://www.w3.org/2006/time#Interval> ;

rdfs:comment "The object property \"hasActualBeginning\" is used to mark the actual beginning of a Phase."@en ;

rdfs:label "has actual beginning"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasActualEnd

occp:hasActualEnd rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasEnd> ;

rdfs:domain <http://www.w3.org/2006/time#Interval> ;

rdfs:comment "The object property \"hasActualEnd\" is used to mark the actual end of a Phase."@en ;

rdfs:label "has actual end"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasActualTime

occp:hasActualTime rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasTime> ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "The object property \"hasActualTime\" is used to mark the actual points in time of an Instant."@en ;

rdfs:label "has actual time"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasCycleNumber

occp:hasCycleNumber rdf:type owl:DatatypeProperty ;

rdfs:domain occp:Cycle ;

rdfs:range xsd:integer ;

rdfs:comment "Specifies the number of iterations in a cycle (e.g., review cycles in CycleA\_PlanningReview)."@en ;

rdfs:label "has cycle number"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasEstimatedBeginning

occp:hasEstimatedBeginning rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasBeginning> ;

rdfs:domain <http://www.w3.org/2006/time#Interval> ;

rdfs:comment "The object property \"hasEstimatedBeginning\" is used to mark the estimated beginning of a Phase."@en ;

rdfs:label "has estimated beginning"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasEstimatedEnd

occp:hasEstimatedEnd rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasEnd> ;

rdfs:domain <http://www.w3.org/2006/time#Interval> ;

rdfs:comment "The object property \"hasEstimatedEnd\" is used to mark the estimated end of a Phase."@en ;

rdfs:label "has estimated end"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasEstimatedTime

occp:hasEstimatedTime rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasTime> ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "The object property \"hasEstimatedTime\" is used to mark the estimated points in time of an Instant."@en ;

rdfs:label "has estimated time"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasInstant

occp:hasInstant rdf:type owl:ObjectProperty ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "The property \"hasInstant\" is used to establish connections between models of constructions or components resp. their corresponding individuals (as representation within an ontology, e.g., a bot:Element) to specific instants of the MsOCCP (e.g., occp:BeginningOfPlanning)."@en ;

rdfs:label "has instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#hasPhase

occp:hasPhase rdf:type owl:ObjectProperty ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "The property \"hasPhase\" is used to connect time instants to certain phases."@en ;

rdfs:label "has phase"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#isInPhase

occp:isInPhase rdf:type owl:ObjectProperty ;

rdfs:comment "The property \"isInPhase\" is used for new life cycles that evolve e.g., from complex repair measures. Example: Part of a building in use is damaged and needs to be replaced. The repair measures need to be planned and so forth and thus start a new life cycle for the replacement element. This new life cycle is taking place within the repair phase of the damaged elements (\"is in Phase of the Repair Phase of said elements\"), as the rest of the building stays intact."@en ;

rdfs:label "is in phase"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#startsCycle

occp:startsCycle rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasBeginning> ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "The object property \"startsCycle\" is used to mark instants that start a new cycle within a phase."@en ;

rdfs:label "starts cycle"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#startsPhase

occp:startsPhase rdf:type owl:ObjectProperty ;

rdfs:subPropertyOf <http://www.w3.org/2006/time#hasBeginning> ;

rdfs:domain <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "The object property \"startsPhase\" is used to mark instants that start a new phase."@en ;

rdfs:label "starts phase"@en .

#################################################################

# Classes

#################################################################

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#BeginOfTenderingProcess

occp:BeginOfTenderingProcess rdf:type owl:Class ;

rdfs:subClassOf occp:Transition ;

rdfs:comment "This instant marks the point in time of the beginning of the tendering process. It is placed between the planning and the construction phase. The tendering process is not considered a phase of the building life cycle, but a transition from the planning to the construction phase. The tendering process starts after the 'Transition' \"completion of planning\"."@en ;

rdfs:label "Begin of Tendering Process"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#BeginningOfPlanning

occp:BeginningOfPlanning rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseA\_Instant ;

rdfs:comment "The Beginning of Planning marks the date of the official start of the planning phase."@en ;

rdfs:label "Beginning of Planning"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#CompletionOfConstruction

occp:CompletionOfConstruction rdf:type owl:Class ;

rdfs:subClassOf occp:Transition ;

rdfs:comment "This instant serves as a transition between the construction and usage phase and marks the point in time of the last acceptance of a building/construction and the finish of the construction works. This instant marks the end of the construction phase."@en ;

rdfs:label "Completion of Construction"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#CompletionOfDeconstruction

occp:CompletionOfDeconstruction rdf:type owl:Class ;

rdfs:subClassOf occp:Transition ;

rdfs:comment "This instant marks the point in time of the completion of the final deconstruction task of a building and marks the end of the life cycle of a building/construction. This instant marks the end of the deconstruction phase."@en ;

rdfs:label "Completion of Deconstruction"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#CompletionOfPlanning

occp:CompletionOfPlanning rdf:type owl:Class ;

rdfs:subClassOf occp:Transition ;

rdfs:comment "This instant serves as a transition from the planning phase to the tendering process and marks the point in time of the last planning review acceptance of a building/construction. This instant marks the end of the planning phase."@en ;

rdfs:label "Completion of Planning"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#ConstructionAcceptance

occp:ConstructionAcceptance rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseC\_Instant ;

rdfs:comment "This instant marks the date of the final acceptance of a construction or component. The acceptance can be granted after the completion if the construction or after the successful elimination of all defects."@en ;

rdfs:label "Construction Acceptance"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#ConstructionCompletion

occp:ConstructionCompletion rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseC\_Instant ;

rdfs:comment "This instant marks the point in time of the completion of construction of a model component of building/construction."@en ;

rdfs:label "Construction Completion"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#ConstructionStart

occp:ConstructionStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseC\_Instant ;

rdfs:comment "This instant marks the point in time of the beginning of the construction works of a model component or building/construction."@en ;

rdfs:label "Construction Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#Cycle

occp:Cycle rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Interval> ;

rdfs:comment "A cycle is a time span with recurring periods, defined by a starting point in time, a period and a duration. They are used to keep track of iterating processes, like multiple reviews during the planning phase, where each rejection of the reviewer causes another planning (and reviewing) cycle."@en ;

rdfs:label "Cycle"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#CycleA\_PlanningReview

occp:CycleA\_PlanningReview rdf:type owl:Class ;

rdfs:subClassOf occp:Cycle ;

rdfs:comment "The CycleA\_PlanningReview starts with the submission of a plan to review and ends with either the approval or the rejection of the plan. If the plan is rejected, a new cycle begins. Only the approval of a plan ends the CycleA\_PlanningReview without starting another."@en ;

rdfs:label "Cycle A Planning Review"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DamageDetection

occp:DamageDetection rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseK\_Instant ;

rdfs:comment "This instant marks the point in time of the detection of a damage that affects a component/construction/building. Multiple events are possible."@en ;

rdfs:label "Damage Detection"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DamageObservation

occp:DamageObservation rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseK\_Instant ;

rdfs:comment "This instant marks the point in time of the observation of a known defect or damage or a component/building/construction. It is usually connected to an inspection, but it could also be used in combination with a damage monitoring system. Multiple events are possible."@en ;

rdfs:label "Damage Observation"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DamageRepair

occp:DamageRepair rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseK\_Instant ;

rdfs:comment "This instant marks the point in time of the repair of a defect or damage of a component/building/construction. It usually is combined with the end of repair measures."@en ;

rdfs:label "Damage Repair"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DataProcurement

occp:DataProcurement rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseA\_Instant ;

rdfs:comment "Marks the date of data procurement measures like geodetic procurement actions or the date a set of data was added to the planning data sets/planning model. Multiple events are possible."@en ;

rdfs:label "Data Procurement"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DeconstructionCompletion

occp:DeconstructionCompletion rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseM\_Instant ;

rdfs:comment "This instant marks the point in time of the completion of the deconstruction works of a component/building/construction. This marks the end of the life cycle of that element."@en ;

rdfs:label "Deconstruction Completion"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DeconstructionStart

occp:DeconstructionStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseM\_Instant ;

rdfs:comment "This instant marks the beginning of the deconstruction works of a component/building/construction."@en ;

rdfs:label "Deconstruction Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DefectElimCompletion

occp:DefectElimCompletion rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseC\_Instant ;

rdfs:comment "This instant marks the point in time of the completion of the contruction works related to the elimination of defects of a model component or building/construction."@en ;

rdfs:label "Defect Elimination Completion"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DefectElimStart

occp:DefectElimStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseC\_Instant ;

rdfs:comment "This instant marks the point in time of the beginning of the contruction works related to the elimination of defects of a model component or building/construction."@en ;

rdfs:label "Defect Elimination Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DesignLifeChange

occp:DesignLifeChange rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseF\_Instant ;

rdfs:comment "This instant marks the point in time of a change of the estimated design life of a model component or building/construction. Different circumstances, such as repair measures or the change in usage, can lead to a change of the design life estimation and multiple dates can mark the change of estimations, thus this instant can be counted. This instant may lay before or after the end of a design life estimation and marks the date of the change of the estimation for the end of the design life, not the beginning of a new start of design life. Multiple events are possible."@en ;

rdfs:label "Design Life Change"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DesignLifeEnd

occp:DesignLifeEnd rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseF\_Instant ;

rdfs:comment "This instant marks the point in time of the end of the estimated design life of a model component or building/construction. Different circumstances, such as repair measures, damages or the change in usage, can lead to a change of the design life estimation and multiple dates can mark the end of the estimated end of the design life, thus multiple entries are possible."@en ;

rdfs:label "Design Life End"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#DesignLifeStart

occp:DesignLifeStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseF\_Instant ;

rdfs:comment "This instant marks the point in time of the start of the estimated design life of a model component or building/construction."@en ;

rdfs:label "Design Life Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#Edit\_ASTR

occp:Edit\_ASTR rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseA\_Instant ;

rdfs:comment "ASTR = after submission to review. This instant is used to mark the date that major or significant changes are made to a planning model after the draft is submitted to the reviewer. Since many changes of this type are possible, multiple entries are possible."@en ;

rdfs:label "Edit ASTR"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#Edit\_BSTR

occp:Edit\_BSTR rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseA\_Instant ;

rdfs:comment "BSTR = Before submission to review. This instant is used to mark the date that major or significant changes are made to a planning model before the draft is submitted to the reviewer. Since many changes of this type are possible, multiple entries are possible."@en ;

rdfs:label "Edit BSTR"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#GeologicalEvent

occp:GeologicalEvent rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseH\_Instant ;

rdfs:comment "This instant marks the point in time of a geological event that affects a component/construction/building, like an earth quake or a subsidence. Multiple events are possible."@en ;

rdfs:label "Geological Event"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#InspectionExecution

occp:InspectionExecution rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseG\_Instant ;

rdfs:comment "This instant marks the point in time of the execution of a regular inspection. Many inspections are possible over the life cycle of a component/building/structure."@en ;

rdfs:label "Inspection Execution"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#IrregularInspection

occp:IrregularInspection rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseG\_Instant ;

rdfs:comment "This instant marks the point in time of the execution of an irregular inspection. Many irregular inspections are possible over the life cycle of a component/building/structure."@en ;

rdfs:label "Irregular Inspection"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#LoadEvent

occp:LoadEvent rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseH\_Instant ;

rdfs:comment "This instant marks the point in time of a irregular load event that affects a component/construction/building. Multiple events are possible."@en ;

rdfs:label "Load Event"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#NewComponent

occp:NewComponent rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseL\_Instant ;

rdfs:comment "This instant marks the point in time of the addition of a new component to an existing component/buildling/construction. For example this can be used to keep track of different parts of a construction, that need to be replaced, such as a new door handle that is fitted to an old door. For bigger changes within existing components/buildings/constructions, it is possible to initiate a sub-life-cycle (from planning until deconstruction), but as this could lead to confusion, this instant can be used to separate the new component (and its life cycle) from the old component, which continues to exist. Multiple events are possible."@en ;

rdfs:label "New Component"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#NextRegularInspection

occp:NextRegularInspection rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseG\_Instant ;

rdfs:comment "This instant marks the point in time of the next of a regular inspection. This date is depending on the inspection interval of the component/building/structure and is usually calculated regarding the date of the last regular inspection. It serves as an orientation or reminder for the planning of the next regular inspection in order to stay within the regular inspection interval. Many inspections are possible over the life cycle of a component/building/structure."@en ;

rdfs:label "Next Regular Inspection"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#Phase

occp:Phase rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Interval> ;

rdfs:comment "Phases describe the main blocks of the life cycle of a building within the chronology. The order is based on the general approach of planning, building, using, maintaining and deconstruction an object in civil engineering. It is a suggestion for a base chronological structure that can be edited and expanded as needed to fit the use case."@en ;

rdfs:label "Phase"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseA\_Instant

occp:PhaseA\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase A - Planning Phase"@en ;

rdfs:label "Phase A Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseA\_Planning

occp:PhaseA\_Planning rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the planning phase. It begins with the beginning of planning and ends with the transition \"CompletionOfPlanning\"."@en ;

rdfs:label "Phase A Planning"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseB\_Instant

occp:PhaseB\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase B - Review Phase"@en ;

rdfs:label "Phase B Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseB\_Review

occp:PhaseB\_Review rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the review phase. It begins with the first submission to review and ends with the transition \"CompletionOfPlanning\"."@en ;

rdfs:label "Phase B Review"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseC\_Construction

occp:PhaseC\_Construction rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the construction phase. It begins with the beginning of construction and ends with the transition \"CompletionOfConstruction\"."@en ;

rdfs:label "Phase C Construction"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseC\_Instant

occp:PhaseC\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase C - Construction Phase"@en ;

rdfs:label "Phase C Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseD\_Instant

occp:PhaseD\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase D - Usage Phase"@en ;

rdfs:label "Phase D Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseD\_Usage

occp:PhaseD\_Usage rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the usage phase. It begins with the beginning of usage and ends with the end of usage."@en ;

rdfs:label "Phase D Usage"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseE\_Instant

occp:PhaseE\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase E - Warranty Phase."@en ;

rdfs:label "Phase E Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseE\_Warranty

occp:PhaseE\_Warranty rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the warranty phase. It begins with the beginning of warranty and ends with the last end of warranty."@en ;

rdfs:label "Phase E Warranty"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseF\_DesignLife

occp:PhaseF\_DesignLife rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the design life phase. It begins with the beginning of design life and ends with the end of design life."@en ;

rdfs:label "Phase F Design Life"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseF\_Instant

occp:PhaseF\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase F - Design Life Phase."@en ;

rdfs:label "Phase F Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseG\_Inspection

occp:PhaseG\_Inspection rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the inspection phase. It begins with the beginning of usage and ends with the beginning of deconstruction."@en ;

rdfs:label "Phase G Inspection"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseG\_Instant

occp:PhaseG\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase G - Inspection Phase."@en ;

rdfs:label "Phase G Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseH\_Instant

occp:PhaseH\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase H - Special Events Phase"@en ;

rdfs:label "Phase H Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseH\_SpecialEvents

occp:PhaseH\_SpecialEvents rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the special events phase. It begins with the first special event and ends with the last."@en ;

rdfs:label "Phase H Special Events"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseK\_Damage

occp:PhaseK\_Damage rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the damage phase. It begins with the first damage detection and ends with the completion of repair or the beginning of deconstruction."@en ;

rdfs:label "Phase K Damage"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseK\_Instant

occp:PhaseK\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase K - Damage Phase."@en ;

rdfs:label "Phase K Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseL\_Instant

occp:PhaseL\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase L - Repair Phase."@en ;

rdfs:label "Phase L Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseL\_Repair

occp:PhaseL\_Repair rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This phase describes the repair phase. It begins with the beginning of repair works and ends with the completion of repair works."@en ;

rdfs:label "Phase L Repair"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseM\_Deconstruction

occp:PhaseM\_Deconstruction rdf:type owl:Class ;

rdfs:subClassOf occp:Phase ;

rdfs:comment "This Phase describes the deconstruction works from the beginning until the end of deconstruction of a component/building/construction."@en ;

rdfs:label "Phase M Deconstruction"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#PhaseM\_Instant

occp:PhaseM\_Instant rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Contains all instants that are within Phase M - Deconstruction Phase."@en ;

rdfs:label "Phase M Instant"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#RepairCompletion

occp:RepairCompletion rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseL\_Instant ;

rdfs:comment "This instant marks the point in time of the completion of damage repair works of a component/construction/building. Multiple events are possible."@en ;

rdfs:label "Repair Completion"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#RepairStart

occp:RepairStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseL\_Instant ;

rdfs:comment "This instant marks the point in time of the beginning of damage repair works of a component/construction/building. Multiple events are possible."@en ;

rdfs:label "Repair Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#ReviewApproval

occp:ReviewApproval rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseB\_Instant ;

rdfs:comment "This instant marks the point in time of the acceptance of a planning model by the reviewer. It also marks the end of the planning and review phases for the reviewed and approved model or model component."@en ;

rdfs:label "Review Approval"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#ReviewRejection

occp:ReviewRejection rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseB\_Instant ;

rdfs:comment "This instant marks the point in time of the rejection of a planning model by the reviewer. Multiple events are possible."@en ;

rdfs:label "Review Rejection"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#ReviewStart

occp:ReviewStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseB\_Instant ;

rdfs:comment "This instant marks the starting point of the reviewing process. Multiple events are possible (in case of a rejection and restart of the review)."@en ;

rdfs:label "Review Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#Submission

occp:Submission rdf:type owl:Class ;

rdfs:subClassOf occp:Transition ;

rdfs:comment "The submission instant marks the point in time of the end of the tendering process and serves as a transition to the construction phase."@en ;

rdfs:label "Submission"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#SubmissionToReview

occp:SubmissionToReview rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseA\_Instant ;

rdfs:comment "This instant marks the date of the submission of a planning draft or model to the reviewer. It starts the reviewing process and thus Phase B. Multiple iterations may be needed to get the reviewers approval."@en ;

rdfs:label "Submission to Review"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#Transition

occp:Transition rdf:type owl:Class ;

rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

rdfs:comment "Transitions mark key milestones between phases or special triggering events regarding a project - not single components. They usually trigger or allow for the beginning of the next consecutive phase or milestone of the project and/or marking the completion of a phase."@en ;

rdfs:label "Transition"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#UsageChange

occp:UsageChange rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseD\_Instant ;

rdfs:comment "This instant marks the point in time of a change in usage of model component or building/construction. Since there can be multiple changes in usage, this instant can be counted. This also means there can be multiple usage phases (with gaps in between them) and thus multiple starts and endings of these usage phases. This instant is after the end of a previous usage phase and before the start of a new usage phase."@en ;

rdfs:label "Usage Change"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#UsageEnd

occp:UsageEnd rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseD\_Instant ;

rdfs:comment "This instant marks the point in time of the end of usage of model component or building/construction. Multiple events are possible."@en ;

rdfs:label "Usage End"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#UsageStart

occp:UsageStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseD\_Instant ;

rdfs:comment "This instant marks the point in time of the beginning of usage of model component or building/construction. Multiple events are possible."@en ;

rdfs:label "Usage Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#WarrantyChange

occp:WarrantyChange rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseE\_Instant ;

rdfs:comment "This instant marks the point in time of a change of the warranty of a model component or building/construction. Repair measures can lead to the restart of the warranty period or the prolonging af the warranty, thus multiple events are possible."@en ;

rdfs:label "Warranty Change"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#WarrantyEnd

occp:WarrantyEnd rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseE\_Instant ;

rdfs:comment "This instant marks the point in time of the end of the warranty period of a model component or building/construction. Repair measures can lead to the restart of the warranty period and multiple dates that mark the end of the warranty."@en ;

rdfs:label "Warranty End"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#WarrantyStart

occp:WarrantyStart rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseE\_Instant ;

rdfs:comment "This instant marks the point in time of the beginning of the warranty period of a model component or building/construction. Repair measures can lead to the restart of the warranty period and multiple dates that mark the beginning of the warranty."@en ;

rdfs:label "Warranty Start"@en .

### http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#WeatherEvent

occp:WeatherEvent rdf:type owl:Class ;

rdfs:subClassOf occp:PhaseH\_Instant ;

rdfs:comment "This instant marks the point in time of a weather event that affects a component/construction/building, like an flood or hurricane. Since multiple events are possible."@en ;

rdfs:label "Weather Event"@en .

#################################################################

# SHACL SHAPES

#################################################################

### GENERAL SHAPE RULES

osh:GeneralDatePropertyShape a sh:PropertyShape ; # All dates are in the xs-date format (YYYY-MM-DD).

sh:path [ sh:alternativePath ( time:hasTime time:hasBeginning occp:hasActualBeginning occp:hasEstimatedBeginning time:hasEnd occp:hasActualEnd occp:hasEstimatedEnd occp:hasNewTime occp:hasPreviousTime ) ] ;

sh:datatype xsd:date ;

sh:message "The date must be in the format YYYY-MM-DD." ;

sh:severity sh:Violation .

osh:CycleShape a sh:NodeShape ;

sh:targetClass occp:Cycle ;

sh:property [

sh:path occp:hasCycleNumber ;

sh:datatype xsd:integer ;

sh:minCount 1 ;

sh:maxCount 1 ;

sh:minInclusive 1 ;

sh:message "Each Cycle must specify exactly one positive integer for the number of iterations via occp:hasCycleNumber." ;

sh:severity sh:Violation ;

] .

### ORDER OF PHASES AND TRANSITIONS

osh:ShapeOfPhaseA a sh:NodeShape ; # Phase A (Planning)

sh:targetClass occp:PhaseA\_Planning ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:or ( # Phase order options

[

sh:path occp:before ; # Phase A usually starts and ends before Phase C

sh:class occp:PhaseC\_Construction ;

]

[

sh:path occp:before ; # Phase A usually starts and ends before Phase B

sh:class occp:PhaseB\_Review ;

sh:message "Wrong assignment of Phase A (Planning). Phase A usually starts and ends before Phases B (Review) and C (Construction)." ;

sh:severity sh:Warning ;

]

) ;

sh:and ( # Phase order options

[

sh:path occp:beginsBefore ; # Phase A must start before Phase B

sh:class occp:PhaseB\_Review ;

]

[

sh:path occp:beginsBefore ; # Phase A must start before Phase C

sh:class occp:PhaseC\_Construction ;

sh:message "Wrong assignment of Phase A (Planning). Phase A must start before Phases B (Review) and C (Construction)." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual beginning) in Phase A (Planning) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning in Phase A (Planning) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) in Phase A (Planning) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase A (Planning) - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfPhaseB a sh:NodeShape ; # Phase B (Review)

sh:targetClass occp:PhaseB\_Review ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:or ( # Phase order options - Phase C (Construction)

[

sh:path occp:before ; # Phase B usually starts and ends before Phase C

sh:class occp:PhaseC\_Construction ;

]

[

sh:path occp:beginsBefore ; # Phase B must start before Phase C (Construction)

sh:class occp:PhaseC\_Construction ;

sh:message "Wrong assignment of Phase B (Planning). Phase B usually starts and ends before Phase C (Construction). Phase B must start before Phase C." ;

sh:severity sh:Violation ;

]

) ;

sh:or ( # Phase order options - Transition A (Completion of Planning)

[

sh:path occp:before ; # Phase B should end before the Completion of Planning

sh:class occp:CompletionOfPlanning ;

]

[

sh:path occp:endsWith ; # Phase B must end with the Completion of Planning

sh:class occp:CompletionOfPlanning ;

sh:message "Wrong assignment Phase B (Review). Phase B must end before or with Transition A (Completion of Planning)." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual beginning) in Phase B (Review) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning in Phase B (Review) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) in Phase B (Review) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase B (Review) - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfTransitionA a sh:NodeShape ; # Transition A (Completion of Planning)

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:targetClass occp:CompletionOfPlanning ;

sh:property [ # Transition order

sh:path occp:before ;

sh:class occp:BeginOfTenderingProcess ;

sh:message "Wrong assignment of Transition A (Completion of Planning) or Transition B (Begin of Tendering Process). Transition A usually occurs before Transition B." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Transition time (single point in time)

sh:path time:hasTime ;

sh:minCount 1 ;

sh:maxCount 1 ;

sh:message "Missing or too many dates for Transition A (Completion of Planning) - max. & min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfTransitionB a sh:NodeShape ; # Transition B (Begin Of Tendering Process)

sh:targetClass occp:BeginOfTenderingProcess ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Transition order

sh:path occp:before ;

sh:class occp:Submission ;

sh:message "Wrong assignment of Transition B (Begin of Tendering Process) or Transition C (Submission). Transition B usually occurs before Transition C." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Transition time (single point in time)

sh:path time:hasTime ;

sh:minCount 1 ;

sh:maxCount 1 ;

sh:message "Missing or too many dates for Transition B (Begin of Tendering Process) - max. & min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfTransitionC a sh:NodeShape ; # Transition C (Submission)

sh:targetClass occp:Submission ;

sh:property [ # Transition order

sh:path occp:before ;

sh:class occp:PhaseC\_Construction ;

sh:message "Wrong assignment of Transition C (Submission) or Phase C (Construction). Transition C usually occurs before Phase C." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Transition time (single point in time)

sh:path time:hasTime ;

sh:maxCount 1 ;

sh:minCount 1 ;

sh:message "Missing or too many dates for Transition C (Submission) - max. & min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfPhaseC a sh:NodeShape ; # Phase C (Construction)

sh:targetClass occp:PhaseC\_Construction ;

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:CompletionOfPlanning ;

sh:message "Wrong assignment of Phase C (Construction) or Transition A (Completion of Planning). Phase C usually starts and ends after Transition A." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:Submission ;

sh:message "Wrong assignment of Phase C (Construction) or Transition C (Submission). Phase C usually starts and ends after Transition C" ;

sh:severity sh:Warning ;

] ;

sh:or ( # Phase order options

[

sh:path occp:before ;

sh:class occp:CompletionOfConstruction ;

]

[

sh:path occp:endsWith ;

sh:class occp:CompletionOfConstruction ;

sh:message "Wrong assignment of Phase C (Construction) or Transition D (Completion of Construction). Phase C must end before or with Transition D." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual beginning) in Phase C (Construction) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning in Phase C (Construction) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) in Phase C (Construction) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase C (Construction) - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfTransitionD a sh:NodeShape ; # Transition D (Completion Of Construction)

sh:targetClass occp:CompletionOfConstruction ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Transition time (single point in time)

sh:path time:hasTime ;

sh:minCount 1 ;

sh:maxCount 1 ;

sh:message "Missing or too many dates for Transition D (Completion of Construction) - max. & min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfPhaseD a sh:NodeShape ; # Phase D (Usage)

sh:targetClass occp:PhaseD\_Usage ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:CompletionOfConstruction ;

sh:message "Wrong assignment Phase D (Usage) or Transition D (Completion of Construction). Phase D usually starts and ends after Transition D." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Phase order

sh:path occp:before ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Wrong assignment Phase D (Usage) or Phase M (Deconstruction). Phase D usually starts and ends before Phase M. We don't want anyone to get hurt, do we?" ;

sh:severity sh:Warning ;

] ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual beginning) in Phase D (Usage) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning in Phase D (Usage) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) in Phase D (Usage) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase D (Usage) - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfPhaseE a sh:NodeShape ; # Phase E (Warranty)

sh:targetClass occp:PhaseE\_Warranty ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:CompletionOfConstruction ;

sh:message "Wrong assignment Phase E (Warranty) or Transition D (Completion of Construction). Phase E should start and end after Transition D." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Begin date

sh:path time:hasBeginning ;

sh:minCount 1 ;

sh:message "Missing date (Beginning) in Phase E (Warranty) - min. count = 1." ;

sh:severity sh:Warning ;

] ;

sh:property [ # End date

sh:path time:hasEnd ;

sh:minCount 1 ;

sh:message "Missing date (End) in Phase E (Warranty) - min. count = 1." ;

sh:severity sh:Warning ;

] .

osh:ShapeOfPhaseF a sh:NodeShape ; # Phase F (DesignLife)

sh:targetClass occp:PhaseF\_DesignLife ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:CompletionOfConstruction ;

sh:message "Wrong assignment of Phase F (Design Life) or Transition D (Completion of Construction). Phase F should start and end after Transition D." ;

sh:severity sh:Warning ;

] ;

sh:or ( # Phase order options - Phase D

[

sh:path occp:beginsBefore ;

sh:class occp:PhaseD\_Usage ;

]

[

sh:path occp:beginsWith ;

sh:class occp:PhaseD\_Usage ;

sh:message "Wrong assignment of Phase F (Design Life) or Phase D (Usage). Phase F should begin before or with Phase D." ;

sh:severity sh:Warning ;

]

) ;

sh:property [

sh:path occp:before ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Wrong assignment of Phase F (Design Life) or Phase M (Deconstruction). Phase F should begin and end before Phase M." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual beginning) in Phase F (Design Life) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning in Phase F (Design Life) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) in Phase F (Design Life) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase F (Design Life) - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfPhaseG a sh:NodeShape ; # Phase G (Inspection)

sh:targetClass occp:PhaseG\_Inspection ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:CompletionOfConstruction ;

sh:message "Wrong assignment Phase G (Inspection) or Transition D (Completion of Construction). Phase G should start and end after Transition D." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Phase order

sh:path occp:before ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Wrong assignment Phase G (Inspection) or Phase M (Deconstruction). Phase G should begin and end before Phase M." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

] ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

] ;

] .

osh:ShapeOfPhaseH a sh:NodeShape ; # Phase H (SpecialEvents) - Special events can occur anytime, so this is to be handled like a super-phase. It can be used to describe or record events that have shaped the landscape like earthquakes in the past, but also events that occure during or after the construction's life cycle.

sh:targetClass occp:PhaseH\_SpecialEvents ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:or ( # A special event is either happening on a single day or point in time or is a time interval.

[

sh:path time:hasTime ; # Event time (single point in time)

]

[

sh:and ( # Special event interval (beginning AND end)

[

sh:path time:hasBeginning ;

]

[

sh:path time:hasEnd ;

]

)

]

) .

osh:ShapeOfPhaseK a sh:NodeShape ; # Phase K (Damage)

sh:targetClass occp:PhaseK\_Damage ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:Submission ;

sh:message "Wrong assignment Phase K (Damage) or Transition C (Submission). Phase K should be after Transition C." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Phase order

sh:path occp:before ;

sh:class occp:CompletionOfDeconstruction ;

sh:message "Wrong assignment Phase K (Damage) or Transition E (Completion of Deconstruction). Phase K should start and end before Transition E." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Begin date

sh:path time:hasBeginning ;

] ;

sh:property [ # End date

sh:path time:hasEnd ;

] .

osh:ShapeOfPhaseL a sh:NodeShape ; # Phase L (Repair)

sh:targetClass occp:PhaseL\_Repair ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:Submission ;

sh:message "Wrong assignment Phase L (Repair) or Transition C (Submission). Phase L should start and end after Transition C." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Phase order

sh:path occp:before ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Wrong assignment Phase L (Repair) or Phase M (Deconstruction). Phase L should start and end before Phase M." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual beginning) in Phase L (Repair) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning in Phase L (Repair) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) in Phase L (Repair) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase L (Repair) - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfPhaseM a sh:NodeShape ; # Phase M (Deconstruction)

sh:targetClass occp:PhaseM\_Deconstruction ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Phase order

sh:path occp:after ;

sh:class occp:CompletionOfConstruction ;

sh:message "Wrong assignment Phase M (Deconstruction) or Transition D (Completion Of Construction). Phase M should start and end after Transition D." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Begin date options

sh:path time:hasBeginning ; # Begin date has to be set (hasBeginning)

sh:qualifiedValueShape [ # The following object properties are valid options to set begin date

sh:path (

occp:hasEstimatedBeginning

occp:hasActualBeginning

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual beginning) in Phase M (Deconstruction) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning in Phase M (Deconstruction) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) in Phase M (Deconstruction) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase M (Deconstruction) - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ShapeOfTransitionE a sh:NodeShape ; # Transition E (Completion Of Deconstruction)

sh:targetClass occp:CompletionOfDeconstruction ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:or ( # Transition order options

[

sh:path occp:after ;

sh:class occp:PhaseM\_Deconstruction ;

]

[

sh:path occp:endsWith ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Wrong assignment of Transition E (Completion Of Deconstruction) or Phase M (Deconstruction). Transition E must occur after or end with Phase M." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Transition time (single point in time)

sh:path time:hasTime ;

sh:minCount 1 ;

sh:maxCount 1 ;

sh:message "Missing or too many date(s) for Transition E (Completion Of Deconstruction) - max. & min. count = 1." ;

sh:severity sh:Violation ;

] .

### ORDER OF INSTANTS

### PHASE A INSTANTS

# PHASE ASSIGNMENT

osh:PhaseAInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseA\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseA\_Planning ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase A (Planning)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE A - PLANNING

osh:BeginningOfPlanningShape a sh:NodeShape ; # Instant Beginning Of Planning (Phase A - Planning)

sh:targetClass occp:BeginningOfPlanning ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant (Beginning Of Planning) starts a new Phase A (Planning).

sh:path occp:startsPhase ;

sh:class occp:PhaseA\_Planning ;

] ;

sh:property [ # Instant (Beginning Of Planning) starts a new Cycle A (Planning/Review).

sh:path occp:startsCycle ;

sh:class occp:CycleA\_PlanningReview ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:before ;

sh:class occp:DataProcurement ;

sh:message "Wrong assignment (chronological order) of Beginning Of Planning or Data Procurement." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Beginning of Planning - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Time stamp - limitation

sh:path occp:hasActualTime ;

sh:maxCount 1 ;

sh:message "Too many dates for Actual Time stamp for Beginning of Planning - max. count = 1" ;

sh:severity sh:Violation ;

] .

osh:DataProcurementShape a sh:NodeShape ; # Instant Data Procurement (Phase A - Planning)

sh:targetClass occp:DataProcurement ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:BeginningOfPlanning ;

sh:message "Wrong assignment (chronological order) of Data Procurement or Beginning Of Planning." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:before ;

sh:class occp:SubmissionToReview ;

sh:message "Wrong assignment (chronological order) of Data Procurement or Submission To Review." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ;

sh:minCount 1 ;

sh:message "Missing date for Data Procurement - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:Edit\_BSTRShape a sh:NodeShape ; # Instant Edit Before Submission To Review (Phase A - Planning)

sh:targetClass occp:Edit\_BSTR ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:BeginningOfPlanning ;

sh:message "Wrong assignment (chronological order) of Edit Before Submission To Review or Beginning Of Planning." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:before ;

sh:class occp:SubmissionToReview ;

sh:message "Wrong assignment (chronological order) of Edit Before Submission To Review or Submission To Review." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ;

sh:minCount 1 ;

sh:message "Missing date for Edit Before Submission To Review - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:SubmissionToReviewShape a sh:NodeShape ; # Instant Submission To Review (Phase A - Planning)

sh:targetClass occp:SubmissionToReview ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:or ( # Instant's chronological order within the Phase - 2 Options

[

sh:path occp:after ;

sh:class occp:Edit\_BSTR ;

]

[

sh:path occp:after ;

sh:class occp:Edit\_ASTR ;

sh:message "Wrong assignment (chronological order) of Submission To Review or Edit Before/After Submission To Review." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Submission to Review - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:Edit\_ASTRShape a sh:NodeShape ; # Instant Edit After Submission To Review (Phase A - Planning)

sh:targetClass occp:Edit\_ASTR ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [

sh:path occp:after ;

sh:qualifiedValueShape [

sh:path (

sh:class occp:SubmissionToReview

sh:class occp:ReviewRejection

) ;

] ;

sh:qualifiedMinCount 1 ;

sh:message "Wrong assignment (Submission To Review/Edit After Submission To Review)." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant (Edit After Submission To Review) starts a new Cycle A (Planning/Review).

sh:path occp:startsCycle ;

sh:class occp:CycleA\_PlanningReview ;

] ;

sh:property [

sh:path time:hasTime ; # Instant needs a date as time stamp.

] .

### CYCLE SHAPE

osh:CycleAShape a sh:NodeShape ;

sh:targetClass occp:CycleA\_PlanningReview ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Actual Begin date - limitation

sh:path occp:hasActualBeginning ;

sh:maxCount 1 ;

sh:message "Too many dates for actual Beginning for Cycle A (Planning and Review) - max. count = 1" ;

sh:severity sh:Violation ;

] ;

sh:property [ # End date options

sh:path time:hasEnd ; # End date has to be set (hasEnd)

sh:qualifiedValueShape [ # The following object properties are valid options to set end date

sh:path (

occp:hasEstimatedEnd

occp:hasActualEnd

) ;

sh:minCount 1 ;

] ;

sh:message "Missing date (estimated or actual end) for Cycle A (Planning and Review) - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual End date - limitation

sh:path occp:hasActualEnd ;

sh:maxCount 1 ;

sh:message "Too many dates for actual End in Phase M (Deconstruction) - max. count = 1." ;

sh:severity sh:Violation ;

] .

### PHASE B INSTANTS

# PHASE ASSIGNMENT

osh:PhaseBInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseB\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseB\_Review ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase B (Review)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE B - REVIEW

osh:ReviewStartShape a sh:NodeShape ; # Instant Review Start (Phase B - Review)

sh:targetClass occp:ReviewStart ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [

sh:path occp:startsPhase ;

sh:class occp:PhaseB\_Review ;

] ;

sh:or ( # Instant's chronological order within the Phase - 2 Options

[

sh:path occp:before ;

sh:class occp:ReviewApproval ;

sh:message "Wrong assignment (chronological order) of Review Start or Review Approval." ;

sh:severity sh:Violation ;

]

[

sh:path occp:before ;

sh:class occp:ReviewRejection ;

sh:message "Wrong assignment (chronological order) of Review Start or Review Rejection." ;

sh:severity sh:Violation ;

]

) ;

sh:property [

sh:path occp:after ;

sh:class occp:SubmissionToReview ;

sh:message "Wrong assignment (chronological order) of Review Start or Submission to Review." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Review Start - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ReviewApprovalShape a sh:NodeShape ; # Instant Review Approval (Phase B - Review)

sh:targetClass occp:ReviewApproval ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [

sh:path occp:endsPhase ;

sh:class occp:PhaseA\_Planning ;

] ;

sh:property [

sh:path occp:endsPhase ;

sh:class occp:PhaseB\_Review ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:ReviewStart ;

sh:message "Wrong assignment (chronological order) of Review Approval or Review Start." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant (Review Approval) ends current Cycle A (Planning/Review).

sh:path occp:endsCycle ;

sh:class occp:CycleA\_PlanningReview ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ;

sh:maxCount 1 ;

sh:message "Too many dates for Review Approval - max. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ReviewRejectionShape a sh:NodeShape ; # Instant Review Rejection (Phase B - Review)

sh:targetClass occp:ReviewRejection ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:ReviewStart ;

sh:message "Wrong assignment (chronological order) of Review Rejection or Review Start." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant (Review Rejection) ends current Cycle A (Planning/Review).

sh:path occp:endsCycle ;

sh:class occp:CycleA\_PlanningReview ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ;

] .

### PHASE C INSTANTS

# PHASE ASSIGNMENT

osh:PhaseCInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseC\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseC\_Construction ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase C (Construction)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE C - CONSTRUCTION

osh:ConstructionStartShape a sh:NodeShape ; # Instant Construction Start (Phase C - Construction)

sh:targetClass occp:ConstructionStart ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [

sh:path occp:startsPhase ;

sh:class occp:PhaseC\_Construction ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:before ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Construction Start or Construction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Construction Start - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Time stamp - limitation

sh:path occp:hasActualTime ;

sh:maxCount 1 ;

sh:message "Too many dates for Actual Time stamp for Construction Start - max. count = 1" ;

sh:severity sh:Violation ;

] .

osh:ConstructionCompletionShape a sh:NodeShape ; # Instant Construction Completion (Phase C - Construction)

sh:targetClass occp:ConstructionCompletion ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:ConstructionStart ;

sh:message "Wrong assignment (chronological order) of Construction Completion or Construction Start." ;

sh:severity sh:Violation ;

] ;

sh:or ( # Instant's chronological order within the Phase - 2 Options

[

sh:path occp:before ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Construction Completion or Construction Acceptance." ;

sh:severity sh:Violation ;

]

[

sh:path occp:before ;

sh:class occp:DefectElimStart ;

sh:message "Wrong assignment (chronological order) of Construction Completion or Defect Elimination Start." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Construction Completion - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Time stamp - limitation

sh:path occp:hasActualTime ;

sh:maxCount 1 ;

sh:message "Too many dates for Actual Time stamp for Construction Completion - max. count = 1" ;

sh:severity sh:Violation ;

] .

osh:DefectElimStartShape a sh:NodeShape ; # Instant Defect Elimination Start (Phase C - Construction)

sh:targetClass occp:DefectElimStart ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Defect Elimination Start or Construction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DefectElimCompletion ;

sh:message "Wrong assignment (chronological order) of Defect Elimination Start or Defect Elimination Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Defect Elimination Start - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:DefectElimCompletionShape a sh:NodeShape ; # Instant Defect Elimination Completion (Phase C - Construction)

sh:targetClass occp:DefectElimCompletion ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:DefectElimStart ;

sh:message "Wrong assignment (chronological order) of Defect Elimination Completion or Defect Elimination Start." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Defect Elimination Completion or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Defect Elimination Completion - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:ConstructionAcceptanceShape a sh:NodeShape ; # Instant Construction Acceptance (Phase C - Construction)

sh:targetClass occp:ConstructionAcceptance ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [

sh:path occp:endsPhase ;

sh:class occp:PhaseC\_Construction ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:before ;

sh:class occp:UsageStart ;

sh:message "Wrong assignment (chronological order) of Construction Acceptance or Usage Start." ;

sh:severity sh:Warning ;

] ;

sh:or ( # Instant's chronological order within the Phase - 2 Options

[

sh:path occp:after ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Construction Acceptance or Construction Completion." ;

sh:severity sh:Violation ;

]

[

sh:path occp:after ;

sh:class occp:DefectElimCompletion ;

sh:message "Wrong assignment (chronological order) of Construction Acceptance or Defect Elimination Completion." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Construction Acceptance - min. count = 1." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Actual Time stamp - limitation

sh:path occp:hasActualTime ;

sh:maxCount 1 ;

sh:message "Too many dates for Actual Time stamp for Construction Acceptance - max. count = 1" ;

sh:severity sh:Violation ;

] .

### PHASE D INSTANTS

# PHASE ASSIGNMENT

osh:PhaseDInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseD\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseD\_Usage ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase D (Usage)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE D - USAGE

osh:UsageChangeShape a sh:NodeShape ; # Instant Usage Change (Phase D - Usage)

sh:targetClass occp:UsageChange ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # The current Usage Phase is ended with a Change of Usage.

sh:path occp:endsPhase ;

sh:class occp:PhaseD\_Usage ;

] ;

sh:property [ # A new Usage Phase is started with a Change of Usage.

sh:path occp:startsPhase ;

sh:class occp:PhaseD\_Usage ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Change of Usage or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:UsageStart ;

sh:message "Wrong assignment (chronological order) of Change of Usage or Usage Start." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Usage Change or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Usage Change." ;

sh:severity sh:Warning ;

] .

osh:UsageEndShape a sh:NodeShape ; # Instant Usage End (Phase D - Usage)

sh:targetClass occp:UsageEnd ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # The current Usage Phase is ended with Usage End

sh:path occp:endsPhase ;

sh:class occp:PhaseD\_Usage ;

] ;

sh:or ( # Instant's chronological order within the Phase - 2 Options

[

sh:path occp:after ;

sh:class occp:UsageStart ;

sh:message "Wrong assignment (chronological order) of Usage End or Usage Start." ;

sh:severity sh:Violation ;

]

[

sh:path occp:after ;

sh:class occp:UsageChange ;

sh:message "Wrong assignment (chronological order) of Usage End or Usage Change." ;

sh:severity sh:Violation ;

]

) ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Usage End or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Usage End - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:UsageStartShape a sh:NodeShape ; # Instant Usage Start (Phase D - Usage)

sh:targetClass occp:UsageEnd ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # A new Usage Phase is started with Usage Start

sh:path occp:startsPhase ;

sh:class occp:PhaseD\_Usage ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Usage Start or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Usage Start or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Usage Start - min. count = 1." ;

sh:severity sh:Violation ;

] .

### PHASE E INSTANTS

# PHASE ASSIGNMENT

osh:PhaseEInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseE\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseE\_Warranty ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase E (Warranty)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE E - WARRANTY

osh:WarrantyChangeShape a sh:NodeShape ; # Instant Warranty Change (Phase E - Warranty)

sh:targetClass occp:WarrantyChange ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # The current Warranty Phase is ended with a Change of Warranty.

sh:path occp:endsPhase ;

sh:class occp:PhaseE\_Warranty ;

] ;

sh:property [ # A new Warranty Phase is started with a Change of Warranty.

sh:path occp:startsPhase ;

sh:class occp:PhaseE\_Warranty ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Change of Warranty or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:WarrantyStart ;

sh:message "Wrong assignment (chronological order) of Change of Warranty or Warranty Start." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Warranty Change or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Warranty Change." ;

sh:severity sh:Warning ;

] .

osh:WarrantyEndShape a sh:NodeShape ; # Instant Warranty End (Phase E - Warranty)

sh:targetClass occp:WarrantyEnd ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # The current Warranty Phase is ended with Warranty End

sh:path occp:endsPhase ;

sh:class occp:PhaseE\_Warranty ;

] ;

sh:or ( # Instant's chronological order within the Phase - 2 Options

[

sh:path occp:after ;

sh:class occp:WarrantyStart ;

sh:message "Wrong assignment (chronological order) of Warranty End or Warranty Start." ;

sh:severity sh:Violation ;

]

[

sh:path occp:after ;

sh:class occp:WarrantyChange ;

sh:message "Wrong assignment (chronological order) of Warranty End or Warranty Change." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Warranty End - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:WarrantyStartShape a sh:NodeShape ; # Instant Warranty Start (Phase E - Warranty)

sh:targetClass occp:WarrantyEnd ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # A new Warranty Phase is started with Warranty Start

sh:path occp:startsPhase ;

sh:class occp:PhaseE\_Warranty ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Warranty Start or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Warranty Start or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Warranty Start - min. count = 1." ;

sh:severity sh:Violation ;

] .

### PHASE F INSTANTS

# PHASE ASSIGNMENT

osh:PhaseFInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseF\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseF\_DesignLife ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase F (Design Life)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE F - DESIGN LIFE

osh:DesignLifeChangeShape a sh:NodeShape ; # Instant Design Life Change (Phase F - Design Life)

sh:targetClass occp:DesignLifeChange ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # The current Design Life Phase is ended with a Change of Design Life.

sh:path occp:endsPhase ;

sh:class occp:PhaseF\_DesignLife ;

] ;

sh:property [ # A new Design Life Phase is started with a Change of Design Life.

sh:path occp:startsPhase ;

sh:class occp:PhaseF\_DesignLife ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Change of Design Life or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order within the Phase

sh:path occp:after ;

sh:class occp:DesignLifeStart ;

sh:message "Wrong assignment (chronological order) of Change of Design Life or Design Life Start." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Design Life Change or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Design Life Change." ;

sh:severity sh:Warning ;

] .

osh:DesignLifeEndShape a sh:NodeShape ; # Instant Design Life End (Phase F - Design Life)

sh:targetClass occp:DesignLifeEnd ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # The current Design Life Phase is ended with Design Life End

sh:path occp:endsPhase ;

sh:class occp:PhaseF\_DesignLife ;

] ;

sh:or ( # Instant's chronological order within the Phase - 2 Options

[

sh:path occp:after ;

sh:class occp:DesignLifeStart ;

sh:message "Wrong assignment (chronological order) of Design Life End or Design Life Start." ;

sh:severity sh:Violation ;

]

[

sh:path occp:after ;

sh:class occp:DesignLifeChange ;

sh:message "Wrong assignment (chronological order) of Design Life End or Design Life Change." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Design Life End - min. count = 1." ;

sh:severity sh:Violation ;

] .

osh:DesignLifeStartShape a sh:NodeShape ; # Instant Design Life Start (Phase F - Design Life)

sh:targetClass occp:DesignLifeEnd ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # A new Design Life Phase is started with Design Life Start

sh:path occp:startsPhase ;

sh:class occp:PhaseF\_DesignLife ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Design Life Start or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Design Life Start or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

sh:minCount 1 ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Design Life Start - min. count = 1." ;

sh:severity sh:Violation ;

] .

### PHASE G INSTANTS

# PHASE ASSIGNMENT

osh:PhaseGInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseG\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseG\_Inspection ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase G (Inspection)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE G - INSPECTION

osh:InspectionExecutionShape a sh:NodeShape ; # Instant Inspection Execution (Phase G - Inspection)

sh:targetClass occp:InspectionExecution ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionAcceptance ;

sh:message "Wrong assignment (chronological order) of Inspection Execution or Construction Acceptance." ;

sh:severity sh:Violation ;

] ;

sh:or ( # Instant's chronological order - 2 Options

[

sh:path occp:after ;

sh:class occp:UsageStart ;

sh:message "Wrong assignment (chronological order) of Inspection Execution or Usage Start." ;

sh:severity sh:Violation ;

]

[

sh:path occp:after ;

sh:class occp:UsageChange ;

sh:message "Wrong assignment (chronological order) of Inspection Execution or Usage Change." ;

sh:severity sh:Violation ;

]

) ;

sh:property [ # Instant's chronological order within phase

sh:path occp:before ;

sh:class occp:NextRegularInspection ;

sh:message "Wrong assignment (chronological order) of Inspection Execution or Next Regular Inspection." ;

sh:severity sh:Warning ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Inspection Execution or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Inspection Execution." ;

sh:severity sh:Violation ;

] .

osh:IrregularInspectionShape a sh:NodeShape ; # Instant Irregular Inspection (Phase G - Inspection)

sh:targetClass occp:IrregularInspection ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:UsageStart ;

sh:message "Wrong assignment (chronological order) of Irregular Inspection or Usage Start." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Irregular Inspection or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Irregular Inspection." ;

sh:severity sh:Violation ;

] .

osh:NextRegularInspectionShape a sh:NodeShape ; # Instant Next Regular Inspection (Phase G - Inspection)

sh:targetClass occp:NextRegularInspection ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:UsageStart ;

sh:message "Wrong assignment (chronological order) of Next Regular Inspection or Usage Start." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Next Regular Inspection or Deconstruction Completion." ;

sh:severity sh:Warning ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Next Regular Inspection." ;

sh:severity sh:Violation ;

] .

### PHASE H INSTANTS

# PHASE ASSIGNMENT

osh:PhaseHInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseH\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseH\_SpecialEvents ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase H (Special Events)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE H - SPECIAL EVENTS

osh:GeologicalEventShape a sh:NodeShape ; # Instant Geological Event (Phase H - Special Events)

sh:targetClass occp:GeologicalEvent ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Geological Event." ;

sh:severity sh:Violation ;

] .

osh:LoadEventShape a sh:NodeShape ; # Instant Load Event (Phase H - Special Events)

sh:targetClass occp:LoadEvent ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Load Event or Construction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Load Event or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Load Event." ;

sh:severity sh:Violation ;

] .

osh:WeatherEventShape a sh:NodeShape ; # Instant Weather Event (Phase H - Special Events)

sh:targetClass occp:WeatherEvent ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Geological Event." ;

sh:severity sh:Violation ;

] .

### PHASE K INSTANTS

# PHASE ASSIGNMENT

osh:PhaseKInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseK\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseK\_Damage ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase K (Damage)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE K - DAMAGE

osh:DamageDetectionShape a sh:NodeShape ; # Instant Damage Detection (Phase K - Damage)

sh:targetClass occp:DamageDetection ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Damage Detection or Construction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Damage Detection or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasActualTime ; # Time stamp has to be set

sh:message "Missing time stamp (Actual Time) for Damage Detection." ;

sh:severity sh:Violation ;

] .

osh:DamageObservationShape a sh:NodeShape ; # Instant Damage Observation (Phase K - Damage)

sh:targetClass occp:DamageObservation ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:DamageDetection ;

sh:message "Wrong assignment (chronological order) of Damage Observation or Damage Detection." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Damage Observation or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasActualTime ; # Time stamp has to be set

sh:message "Missing time stamp (Actual Time) for Damage Detection." ;

sh:severity sh:Violation ;

] .

osh:DamageRepairShape a sh:NodeShape ; # Instant Damage Repair (Phase K - Damage)

sh:targetClass occp:DamageRepair ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:DamageDetection ;

sh:message "Wrong assignment (chronological order) of Damage Repair or Damage Detection." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Damage Repair or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Damage Repair." ;

sh:severity sh:Violation ;

] .

### PHASE L INSTANTS

# PHASE ASSIGNMENT

osh:PhaseLInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseL\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseL\_Repair ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase L (Repair)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE L - REPAIR

osh:NewComponentShape a sh:NodeShape ; # Instant New Component (Phase L - Repair)

sh:targetClass occp:NewComponent ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of New Component or Construction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of New Component or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for New Component." ;

sh:severity sh:Violation ;

] .

osh:RepairCompletionShape a sh:NodeShape ; # Instant Repair Completion (Phase L - Repair)

sh:targetClass occp:RepairCompletion ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Repair Completion or Construction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:RepairStart ;

sh:message "Wrong assignment (chronological order) of Repair Completion or Repair Start." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Repair Completion or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Repair Completion." ;

sh:severity sh:Violation ;

] .

osh:RepairStartShape a sh:NodeShape ; # Instant Repair Start (Phase L - Repair)

sh:targetClass occp:RepairStart ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:ConstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Repair Start or Construction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:DamageDetection ;

sh:message "Wrong assignment (chronological order) of Repair Start or Damage Detection." ;

sh:severity sh:Violation ;

] ;

sh:property [

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Repair Start or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Repair Start." ;

sh:severity sh:Violation ;

] .

### PHASE M INSTANTS

# PHASE ASSIGNMENT

osh:PhaseMInstantShape a sh:NodeShape ;

sh:targetClass occp:PhaseM\_Instant ;

sh:property [

sh:path occp:hasPhase ;

sh:class occp:PhaseM\_Deconstruction ;

sh:message "Wrong Phase/Instant assignment - Instant should belong to Phase M (Deconstruction)." ;

sh:severity sh:Violation ;

] .

### INSTANT SHAPES PHASE M - DECONSTRUCTION

osh:DeconstructionCompletionShape a sh:NodeShape ; # Instant Deconstruction Completion (Phase M - Deconstruction)

sh:targetClass occp:DeconstructionCompletion ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:DeconstructionStart ;

sh:message "Wrong assignment (chronological order) of Deconstruction Completion or Deconstruction Start." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Deconstruction Completion." ;

sh:severity sh:Violation ;

] .

osh:DeconstructionStartShape a sh:NodeShape ; # Instant Deconstruction Start (Phase M - Deconstruction)

sh:targetClass occp:DeconstructionCompletion ;

sh:property osh:GeneralDatePropertyShape ; # Date format YYYY-MM-DD

sh:property [ # Instant's chronological order

sh:path occp:after ;

sh:class occp:UsageEnd ;

sh:message "Wrong assignment (chronological order) of Deconstruction Start or Usage End." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant's chronological order

sh:path occp:before ;

sh:class occp:DeconstructionCompletion ;

sh:message "Wrong assignment (chronological order) of Deconstruction Start or Deconstruction Completion." ;

sh:severity sh:Violation ;

] ;

sh:property [ # Instant has a date as time stamp.

sh:path time:hasTime ; # Time stamp has to be set

sh:qualifiedValueShape [ # The following object properties are valid options to set time stamp - no minCount.

sh:path (

occp:hasEstimatedTime

occp:hasActualTime

) ;

] ;

sh:message "Missing time stamp (Estimated or Actual Time) for Deconstruction Start." ;

sh:severity sh:Violation ;

] .

### Current End of Ontology

## OULD – Ontology for Updates and Linked Data - TBox

@prefix oush: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD/shape#> .

@prefix ould: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#> .

@prefix occp: <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix sh: <http://www.w3.org/ns/shacl#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@base <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#> .

<http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD> rdf:type owl:Ontology ;

                                                                        owl:imports <http://www.w3.org/2006/time#2016> ;

                                                                        owl:imports <http://www.semanticweb.org/albrechtvaatz/ontologies/2022/9/cMod\_V0.1#> ;

                                                                        rdfs:comment "First workaround to implement OCCP-based updates and version control features."@en .

#################################################################

#    Object Properties

#################################################################

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasLinkedData

    ould:hasLinkedData rdf:type owl:ObjectProperty ;

                       rdfs:domain ould:UpdatableEntity ;

                       rdfs:range ould:LinkedData ;

                       rdfs:comment "Links an IFC component to an external dataset, document, or other reference stored either online or locally (see also: ould:hasOnlineData and ould:hasOfflineData for online/offline specifics)."@en ;

                       rdfs:label "has linked data"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasUpdate

    ould:hasUpdate rdf:type owl:ObjectProperty ;

                   rdfs:domain ould:UpdatableEntity ;

                   rdfs:range ould:Update ;

                   rdfs:comment "Connects an UpdatableEntity to an existing Update. If a prior Update exists, a new UpdateChain is initiated to track subsequent modifications."@en ;

                   rdfs:label "has update"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasPredecessor

    ould:hasPredecessor rdf:type owl:ObjectProperty ;

                        rdfs:subPropertyOf ould:hasUpdate ;

                        rdfs:domain ould:Update ;

                        rdfs:range ould:Update ;

                        owl:inverseOf ould:hasSuccessor ;

                        rdf:type owl:TransitiveProperty ;

                        rdfs:comment "Links an update to its predecessor, enabling chronological tracking of entity changes (e.g., replacements or modifications). The predecessor is an update entry recording the prior state."@en ;

                        rdfs:label "has predecessor"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasSuccessor

    ould:hasSuccessor rdf:type owl:ObjectProperty ;

                      rdfs:subPropertyOf ould:hasUpdate ;

                      rdfs:domain ould:Update ;

                      rdfs:range ould:Update;

                      owl:inverseOf ould:hasPredecessor ;

                      rdf:type owl:TransitiveProperty ;

                      rdfs:comment "Links an update to a subsequent update, enabling chronological tracking of entity changes (e.g., replacements or modifications)."@en ;

                      rdfs:label "has successor"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasUpdatedValue

    ould:hasUpdatedValue rdf:type owl:ObjectProperty ;

                         rdfs:domain ould:UpdatableEntity ;

                         rdfs:range owl:Thing ;

                         rdfs:comment "The object property \"hasUpdatedValue\" links to an update to any modified value (e.g., hash, timestamp, IFC-ID, linked data, or temporal entity) recorded in an MsOCCP context."@en ;

                         rdfs:label "has updated value"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasUpdatedEvent

    ould:hasUpdatedEvent rdf:type owl:ObjectProperty ;

                         rdfs:subPropertyOf ould:hasUpdatedValue ;

                         rdfs:domain ould:UpdatableEntity ;

                         rdfs:range <http://www.w3.org/2006/time#TemporalEntity> ;

                         rdfs:comment "The object property \"hasUpdatedEvent\" links a temporal entity (phase, cycle, transition, or instant) to an update typically recorded in an MsOCCP (Model-specific OCCP) file. It enables chronological tracking of modifications to lifecycle data (e.g., updates to time:hasTime or occp:hasEstimatedEnd), ensuring the integrity of time-based records."@en ;

                         rdfs:label "has updated event"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasNewValue

    ould:hasNewValue rdf:type owl:ObjectProperty ;

                     rdfs:subPropertyOf ould:hasUpdatedValue ;

                     rdfs:domain ould:UpdatableEntity ;

                     rdfs:range owl:Thing ;

                     owl:inverseOf ould:hasPreviousValue ;

                     rdfs:comment "Links an update to the newly recorded value of a modified property."@en ;

                     rdfs:label "has new value"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasPreviousValue

    ould:hasPreviousValue rdf:type owl:ObjectProperty ;

                          rdfs:subPropertyOf ould:hasUpdatedValue ;

                          rdfs:domain ould:UpdatableEntity ;

                          rdfs:range owl:Thing ;

                          owl:inverseOf ould:hasNewValue ;

                          rdfs:comment "Links an update to the previous recorded value of a modified property."@en ;

                          rdfs:label "has previous value"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasPreviousChain

    ould:hasPreviousChain rdf:type owl:ObjectProperty ;

                          owl:inverseOf ould:hasNextChain ;

                          rdf:type owl:TransitiveProperty ;

                          rdfs:domain ould:UpdateChain ;

                          rdfs:range ould:UpdateChain ;

                          rdfs:comment "The object property \"hasPreviousChain\" links an UpdateChain instance to its predecessor, forming a chronological sequence of modifications. Due to its transitive nature, it enables deep tracking of changes across multiple update cycles, e.g., linking sequential modifications of an IFC component."@en ;

                          rdfs:label "has previous chain"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasNextChain

    ould:hasNextChain rdf:type owl:ObjectProperty ;

                      owl:inverseOf ould:hasPreviousChain ;

                      rdf:type owl:TransitiveProperty ;

                      rdfs:domain ould:UpdateChain ;

                      rdfs:range ould:UpdateChain ;

                      rdfs:comment "The object property \"hasNextChain\" links an UpdateChain instance   to its successor, enabling the structured progression of modification records. Its transitivity ensures complete traceability over multiple updates."@en ;

                      rdfs:label "has next chain"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasUpdateChain

    ould:hasUpdateChain rdf:type owl:ObjectProperty ;

                        rdfs:domain ould:UpdatableEntity ;

                        rdfs:range ould:UpdateChain ;

                        rdfs:comment "The object property \"hasUpdateChain\" links an UpdatableEntity to an \"UpdateChain\", grouping multiple related Updates (in contrast to ould:hasUpdate for single Updates)."@en ;

                        rdfs:label "has update chain"@en .

#################################################################

#    Data properties

#################################################################

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasOnlineData

    ould:hasOnlineData rdf:type owl:DatatypeProperty ;

                       rdfs:subPropertyOf ould:hasLinkedData ;

                       rdfs:domain ould:UpdatableEntity ;

                       rdfs:range xsd:anyURI ;

                       rdfs:comment "Links an IFC component to an external dataset, document, or other reference stored online."@en ;

                       rdfs:label "has online data"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasOfflineData

    ould:hasOfflineData rdf:type owl:DatatypeProperty ;

                        rdfs:subPropertyOf ould:hasLinkedData ;

                        rdfs:domain ould:UpdatableEntity ;

                        rdfs:range xsd:string ;

                        rdfs:comment "Links an IFC component to an external dataset, document, or other reference stored offline."@en ;

                        rdfs:label "has offline data"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasDataLocation

    ould:hasDataLocation rdf:type owl:DatatypeProperty ;

                         rdfs:domain ould:LinkedData ;

                         rdfs:range xsd:string ;

                         rdfs:comment "The data property \"hasDataLocation\" specifies the location of the linked data and is used in conjunction with ould:hasOnlineData or ould:hasOfflineData to specify the exact location. It can be a URI (for online storage) or a local file path (for offline storage)."@en ;

                         rdfs:label "has data location"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasHash

    ould:hasHash rdf:type owl:DatatypeProperty ;

                 rdfs:domain ould:UpdatableEntity ;

                 rdfs:range xsd:string ;

                 rdfs:comment "The data property \"hasHash\" is used to store the hash value of an IFC component to record the current state of a IFC component. The hash value is also used to check for changes of the IFC component."@en ;

                 rdfs:label "has hash"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasIFCID

    ould:hasIFCID rdf:type owl:DatatypeProperty ;

                  rdfs:domain ould:UpdatableEntity ;

                  rdfs:range xsd:string ;

                  rdfs:comment "The data property \"hasIFCID\" is used to store the IFC Identifier (e.g., the IFC component's GlobalID) of a component to record and establish a connection between an IFC component and the MsOCCP (Model-specific OCCP)."@en ;

                  rdfs:label "has IFC ID"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasNewHash

    ould:hasNewHash rdf:type owl:DatatypeProperty ;

                    rdfs:subPropertyOf ould:hasHash ;

                    rdfs:domain ould:UpdatableEntity ;

                    rdfs:range xsd:string ;

                    rdfs:comment "The data property \"hasNewHash\" is used to update an existing hash value of an IFC component to record and establish a connection between the old and the new hash. It is used in combination with ould:hasPreviousHash."@en ;

                    rdfs:label "has new hash"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasNewIFCID

    ould:hasNewIFCID rdf:type owl:DatatypeProperty ;

                    rdfs:subPropertyOf ould:hasIFCID ;

                    rdfs:domain ould:UpdatableEntity ;

                    rdfs:range xsd:string ;

                    rdfs:comment "The data property \"hasNewIFCID\" is used to store the IFC new Identifier (e.g., the IFC component's GlobalID) of a component to record and establish a connection between the old and the new ID. It is used in combination with ould:hasPreviousIFCID."@en ;

                    rdfs:label "has new IFC ID"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasNewTime

    ould:hasNewTime rdf:type owl:DatatypeProperty ;

                    rdfs:subPropertyOf <http://www.w3.org/2006/time#hasTime> ;

                    rdfs:domain [ rdf:type owl:Class ;

                                    owl:unionOf ( ould:Update time:Instant ) ] ;

                    rdfs:range xsd:date ;

                    rdfs:comment "The data property \"hasNewTime\" is used to record the updated timestamp of an event in an MsOCCP (Model-specific OCCP) file. It ensures that every change to a temporal entity is chronologically documented, allowing for a complete reconstruction of modifications and their sequence."@en ;

                    rdfs:label "has new time"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasPreviousTime

    ould:hasPreviousTime rdf:type owl:DatatypeProperty ;

                         rdfs:subPropertyOf <http://www.w3.org/2006/time#hasTime> ;

                         rdfs:domain [ rdf:type owl:Class ;

                                    owl:unionOf ( ould:Update time:Instant ) ] ;

                         rdfs:range xsd:date ;

                         rdfs:comment "The data property \"hasPreviousTime\" is used to record the previously set timestamp (created using the original entry property, e.g., time:hasTime) of an existing and prerecorded event (of which the temporal information needs to be changed/corrected) in an MsOCCP (Model-specific OCCP) file. It can be used to keep track of the chronology of entries, allowing for a complete reconstruction of modifications and their sequence."@en ;

                         rdfs:label "has previous time"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasPreviousHash

    ould:hasPreviousHash rdf:type owl:DatatypeProperty ;

                        rdfs:subPropertyOf ould:hasHash ;

                        rdfs:domain ould:UpdatableEntity ;

                        rdfs:range xsd:string ;

                        rdfs:comment "The data property \"hasPreviousHash\" is used to store the old hash value of an IFC component to record and establish a connection between the old and the new hash. It is used in combination with ould:hasNewHash."@en ;

                        rdfs:label "has previous hash"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#hasPreviousIFCID

    ould:hasPreviousIFCID rdf:type owl:DatatypeProperty ;

                        rdfs:subPropertyOf ould:hasIFCID ;

                        rdfs:domain ould:UpdatableEntity ;

                        rdfs:range xsd:string ;

                        rdfs:comment "The data property \"hasPreviousIFCID\" is used to store the old IFC Identifier (e.g., the IFC component's GlobalID) of a component to record and establish a connection between the old and the new ID. It is used in combination with ould:hasNewIFCID."@en ;

                        rdfs:label "has previous IFC ID"@en .

#################################################################

#    Classes

#################################################################

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#LinkedData

    ould:LinkedData rdf:type owl:Class ;

                    rdfs:comment "Represents an external dataset, document, or reference that is linked to an IFC component. This class enables structured integration of additional information into the MsOCCP."@en ;

                    rdfs:label "Linked Data"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#UpdatableEntity

    ould:UpdatableEntity rdf:type owl:Class ;

                         rdfs:comment "A superclass representing any entity that can undergo updates. This includes IFC components, MsOCCP records, temporal entities and other objects that require chronological modification tracking, e.g., attached or linked data."@en ;

                         rdfs:label "Updatable Entity"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#Update

    ould:Update rdf:type owl:Class ;

                rdfs:subClassOf <http://www.w3.org/2006/time#Instant> ;

                rdfs:comment "The class \"Update\" represents a specific modification recorded in an MsOCCP (Model-specific OCCP) file. Each instance captures a single event's transformation, including changes in timestamps, IFC component identifiers, and hash values. By linking to previous updates, it enables full traceability of modifications over time."@en ;

                rdfs:label "Update"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#UpdateChain

    ould:UpdateChain rdf:type owl:Class ;

                rdfs:comment "The class \"UpdateChain\" represents a structured sequence of consecutive updates applied to a specific IFC component or dataset. It groups related updates together, enabling an organized way to track modifications over time. Each \"UpdateChain\" instance is linked to previous and next chains, ensuring a complete versioning history. An \"UpdateChain\" is limited to 21 ould:UpdateChainElement(s) for performance reasons."@en ;

                rdfs:label "Update Chain"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#UpdateChainElement

    ould:UpdateChainElement rdf:type owl:Class ;

                rdfs:subClassOf ould:Update ;

                rdfs:comment "The class \"UpdateChainElement\" represents an update that is part of an \"UpdateChain\". It maintains sequential tracking of updates, ensuring chronological integrity of changes to IFC components and other time-bound entities."@en ;

                rdfs:label "Update Chain Element"@en .

    ###  http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#UpdateReplacement

    ould:UpdateReplacement rdf:type owl:Class ;

                        rdfs:subClassOf ould:Update ;

                        rdfs:comment "Represents an update that replaces one or more existing entities with a new one (e.g., replacing IFC component ifc:Wall123 with ifc:Wall456). It links the predecessor update or entry in the MsOCCP (recording the old state) via ould:hasPredecessor and tracks changes in identifiers and hash values using ould:hasPreviousIFCID, ould:hasNewIFCID, ould:hasPreviousHash, and ould:hasNewHash."@en ;

                        rdfs:label "Update Replacement"@en .

#################################################################

#    SHACL SHAPES

#################################################################

### GENERAL SHAPE RULES

    oush:GeneralDatePropertyShape a sh:PropertyShape ;     # All dates are in the xs-date format (YYYY-MM-DD).

        sh:path [ sh:alternativePath ( time:hasTime time:hasBeginning occp:hasActualBeginning occp:hasEstimatedBeginning time:hasEnd occp:hasActualEnd occp:hasEstimatedEnd ould:hasNewTime ould:hasPreviousTime ) ] ;

        sh:datatype xsd:date ;

        sh:message "The date must be in the format YYYY-MM-DD." ;

        sh:severity sh:Violation .

### UPDATE SHAPE RULES

    oush:UpdateShape a sh:NodeShape ;

        sh:targetClass ould:Update ;

        sh:property oush:GeneralDatePropertyShape ;  # Date format YYYY-MM-DD

        sh:property [

            sh:path time:hasTime ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each Update must have exactly one time stamp (time:hasTime) - min./max. count = 1." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

        sh:path ould:hasNewTime ;

        sh:datatype xsd:date ;

        sh:maxCount 1 ;

        sh:minCount 0 ;  # Optional

        sh:message "Each Update may have at most one ould:hasNewTime timestamp." ;

        sh:severity sh:Violation

        ] ;

        sh:property [

            sh:path ould:hasPreviousTime ;

            sh:datatype xsd:date ;

            sh:maxCount 1 ;

            sh:minCount 0 ;  # Optional

            sh:message "Each Update may have at most one ould:hasPreviousTime value." ;

            sh:severity sh:Violation

        ] ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "If hasNewTime is specified, hasPreviousTime must also be provided, and vice versa." ;

            sh:severity sh:Violation ;

            sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ] ;

            sh:select """

                SELECT $this

                WHERE {

                    { $this ould:hasNewTime ?newTime .

                    FILTER NOT EXISTS { $this ould:hasPreviousTime ?prevTime . } }

                    UNION

                    { $this ould:hasPreviousTime ?prevTime .

                    FILTER NOT EXISTS { $this ould:hasNewTime ?newTime . } }

                }

            """

        ] ;

        sh:property [

            sh:path ould:hasUpdatedEvent ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each Update must reference exactly one updated event (time:instant or time:interval)." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasIFCID ;

            sh:minCount 0 ;

            sh:maxCount 1 ;

            sh:datatype xsd:string ;

            sh:message "Each Update can only be linked to one IFC component via ould:hasIFCID." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasHash ;

            sh:minCount 0 ;

            sh:maxCount 1 ;

            sh:datatype xsd:string ;

            sh:message "Each Update can be linked to at most one IFC component's hash value via ould:hasHash." ;

            sh:severity sh:Violation ;

        ] ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "hasNewTime must be later than hasPreviousTime if both exist." ;

            sh:severity sh:Violation ;

            sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ] ;

            sh:select """

                SELECT $this ?prevTime ?newTime

                WHERE {

                    $this ould:hasPreviousTime ?prevTime .

                    $this ould:hasNewTime ?newTime .

                    FILTER (?newTime <= ?prevTime)

                }

            """ ;

        ] .

    oush:UpdateChainShape a sh:NodeShape ;

        sh:targetClass ould:UpdateChain ;

        sh:property [

            sh:path ould:hasNextChain ;

            sh:class ould:UpdateChain ;

            sh:maxCount 1 ;

            sh:message "Each UpdateChain may be linked to at most one next chain." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasPreviousChain ;

            sh:class ould:UpdateChain ;

            sh:maxCount 1 ;

            sh:message "Each UpdateChain may be linked to at most one previous chain." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasUpdate ;

            sh:class ould:Update ;

            sh:minCount 1 ;

            sh:maxCount 21 ;

            sh:message "Each UpdateChain must contain between 1 and 21 Updates or UpdateChainElements." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path time:hasBeginning ;

            sh:datatype xsd:date ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each UpdateChain must have a beginning in format xsd:date." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path time:hasEnd ;

            sh:datatype xsd:date ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each UpdateChain must have a End in format xsd:date." ;

            sh:severity sh:Violation ;

        ] ;

        sh:sparql [

        a sh:SPARQLConstraint ;

        sh:message "time:hasBeginning must match the earliest time:hasTime of its Updates." ;

        sh:severity sh:Violation ;

        sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ;

                      sh:declare [ sh:prefix "time" ; sh:namespace "http://www.w3.org/2006/time#" ] ] ;

        sh:select """

            SELECT $this ?begin ?earliest

            WHERE {

                $this time:hasBeginning ?begin .

                $this ould:hasUpdate ?update .

                ?update time:hasTime ?updateTime .

                {

                    SELECT $this (MIN(?time) AS ?earliest)

                    WHERE {

                        $this ould:hasUpdate ?u .

                        ?u time:hasTime ?time .

                    }

                    GROUP BY $this

                }

                FILTER (?begin != ?earliest)

            }

        """ ;

        ] ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "time:hasEnd must match the latest time:hasTime of its Updates." ;

            sh:severity sh:Violation ;

            sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ;

                        sh:declare [ sh:prefix "time" ; sh:namespace "http://www.w3.org/2006/time#" ] ] ;

            sh:select """

                SELECT $this ?end ?latest

                WHERE {

                    $this time:hasEnd ?end .

                    $this ould:hasUpdate ?update .

                    ?update time:hasTime ?updateTime .

                    {

                        SELECT $this (MAX(?time) AS ?latest)

                        WHERE {

                            $this ould:hasUpdate ?u .

                            ?u time:hasTime ?time .

                        }

                        GROUP BY $this

                    }

                    FILTER (?end != ?latest)

                }

            """ ;

        ] ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "An UpdateChain must contain all Updates of its UpdatableEntity once it exists." ;

            sh:severity sh:Violation ;

            sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ] ;

            sh:select """

                SELECT $this ?entity ?update

                WHERE {

                    ?entity ould:hasUpdateChain $this .

                    ?entity ould:hasUpdate ?update .

                    FILTER NOT EXISTS { $this ould:hasUpdate ?update . }

                    {

                        SELECT ?entity (COUNT(?u) AS ?updateCount)

                        WHERE {

                            ?entity ould:hasUpdate ?u .

                        }

                        GROUP BY ?entity

                        HAVING (?updateCount > 1)

                    }

                }

            """ ;

        ] .

    oush:UpdateChainContinuityConstraint a sh:NodeShape ;

        sh:targetClass ould:UpdatableEntity ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "If an UpdatableEntity has more than 21 Updates in a single chain, they must be split into chains with max 21 Updates each, linked via hasNextChain." ;

            sh:severity sh:Violation ;

            sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ] ;

            sh:select """

                SELECT $this ?chain ?updateCount

                WHERE {

                    $this ould:hasUpdateChain ?chain .

                    {

                        SELECT $this ?chain (COUNT(?u) AS ?updateCount)

                        WHERE {

                            ?chain ould:hasUpdate ?u .

                        }

                        GROUP BY $this ?chain

                        HAVING (?updateCount > 21)

                    }

                }

            """

        ] .

    oush:UpdateChainElementShape a sh:NodeShape ;

        sh:targetClass ould:UpdateChainElement ;

        sh:property [

            sh:path time:hasTime ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each UpdateChainElement must have exactly one time:hasTime timestamp." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasUpdatedEvent ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each UpdateChainElement must reference exactly one updated event (time:instant or time:interval)." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasIFCID ;

            sh:minCount 0 ;

            sh:maxCount 2 ;

            sh:message "Each UpdateChainElement can be linked to max. two IFC components via ould:hasIFCID." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasHash ;

            sh:minCount 0 ;

            sh:maxCount 2 ;

            sh:message "Each UpdateChainElement can have max. two recorded hash values via ould:hasHash." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasNewTime ;

            sh:datatype xsd:date ;

            sh:minCount 0 ;

            sh:maxCount 1 ;

            sh:message "Each UpdateChainElement can have at most one ould:hasNewTime timestamp in xsd:date format." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasPreviousTime ;

            sh:maxCount 1 ;

            sh:message "An UpdateChainElement may have at most one ould:hasPreviousTime value." ;

            sh:severity sh:Warning ;

        ] .

    oush:UpdateChainContinuityConstraint a sh:NodeShape ;

        sh:targetClass ould:UpdatableEntity ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "If an UpdatableEntity has more than 21 Updates, they must be split into chains with max 21 Updates each, linked via hasNextChain. After split, the old chain must have max 20 Updates." ;

            sh:severity sh:Violation ;

            sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ] ;

            sh:select """

                SELECT $this ?chain ?updateCount

                WHERE {

                    $this ould:hasUpdate ?update .

                    $this ould:hasUpdateChain ?chain .

                    {

                        SELECT $this ?chain (COUNT(?u) AS ?updateCount)

                        WHERE {

                            $this ould:hasUpdate ?u .

                            ?chain ould:hasUpdate ?u .

                        }

                        GROUP BY $this ?chain

                        HAVING (?updateCount > 21)

                    }

                }

            """

        ] .

    oush:UpdatableEntityBaselineShape a sh:NodeShape ;

        sh:targetClass ould:UpdatableEntity ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "Each UpdatableEntity must have at least one Update with a hasIFCID and hasHash." ;

            sh:severity sh:Violation ;

            sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ] ;

            sh:select """

                SELECT $this

                WHERE {

                    $this ould:hasUpdate ?update .

                    FILTER NOT EXISTS { ?update ould:hasIFCID ?ifcid . }

                    FILTER NOT EXISTS { ?update ould:hasHash ?hash . }

                }

            """ ;

        ] .

    oush:UpdateReplacementShape a sh:NodeShape ;

        sh:targetClass ould:UpdateReplacement ;

        sh:property oush:GeneralDatePropertyShape ;  # Date format YYYY-MM-DD

        sh:property [

            sh:path time:hasTime ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each UpdateReplacement must have exactly one timestamp (time:hasTime)." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasPredecessor ;

            sh:class ould:Update ;

            sh:minCount 1 ;

            sh:message "Each UpdateReplacement must reference at least one predecessor Update via ould:hasPredecessor." ;

            sh:severity sh:Violation ;

        ] ;

        sh:property [

            sh:path ould:hasSuccessor ;

            sh:class ould:Update ;

            sh:minCount 0 ;

            sh:maxCount 1 ;

            sh:message "An UpdateReplacement may have at most one successor (e.g., ould:Update or ould:UpdateReplacement)." ;

            sh:severity sh:Violation ;

        ] .

    oush:ReplacementChronologyConstraint a sh:NodeShape ;

        sh:targetClass ould:UpdateReplacement ;

        sh:sparql [

            a sh:SPARQLConstraint ;

            sh:message "The successor UpdateReplacement must have a later timestamp than its predecessor Update." ;

            sh:severity sh:Violation ;

            sh:prefixes [

                sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ;

            ] ;

            sh:select """

                SELECT $this ?prevTime ?newTime

                WHERE {

                    $this ould:hasPredecessor ?prevUpdate .

                    ?prevUpdate time:hasTime ?prevTime .

                    $this time:hasTime ?newTime .

                    FILTER (?prevTime > ?newTime)

                }

            """ ;

        ] .

    oush:PredecessorConstraint a sh:NodeShape ;

        sh:targetSubjectsOf ould:hasPredecessor ;

        sh:property [

            sh:path ould:hasPredecessor ;

            sh:class ould:Update ;

            sh:minCount 1 ;

            sh:message "Each ould:UpdateReplacement must reference at least one predecessor ould:Update." ;

            sh:severity sh:Violation ;

        ] .

    oush:SuccessorConstraint a sh:NodeShape ;

        sh:targetSubjectsOf ould:hasSuccessor ;

        sh:property [

            sh:path ould:hasSuccessor ;

            sh:class ould:Update ;

            sh:minCount 0 ;

            sh:maxCount 1 ;

            sh:message "Each ould:Update may reference at most one successor (e.g., ould:Update or ould:UpdateReplacement)." ;

            sh:severity sh:Violation ;

        ] .

    oush:UpdateEntityShape a sh:NodeShape ;

        sh:targetSubjectsOf ould:hasUpdate ;

        sh:property [

            sh:path ould:hasUpdate ;

            sh:class ould:Update ;

            sh:minCount 1 ;

            sh:message "Each entity with an update must be linked to at least one ould:Update instance." ;

            sh:severity sh:Violation ;

        ] .

    oush:UpdatedValueShape a sh:NodeShape ;

        sh:targetClass ould:Update ;

        sh:property [

            sh:path ould:hasUpdatedValue ;

            sh:minCount 1 ;

            sh:message "Each update must reference at least one modified entity (e.g., IFC component, phase, or instant)." ;

            sh:severity sh:Violation ;

        ] .

    oush:UpdatedEventShape a sh:NodeShape ;

        sh:targetClass ould:Update ;

        sh:property [

            sh:path ould:hasUpdatedEvent ;

            sh:class <http://www.w3.org/2006/time#TemporalEntity> ;

            sh:minCount 1 ;

            sh:maxCount 1 ;

            sh:message "Each update must be linked to exactly one modified temporal entity (e.g., phase, cycle, transition, or instant)." ;

            sh:severity sh:Violation ;

        ] .

# cMM – Development

## SHACL Validation

Für die Validierung (sowohl jetzt im Entwicklungsstadium zur Vorbereitung der Veröffentlichung der Ontologien als auch für die Echtzeitvalidierung von MsOCCP in der cMM) habe ich ein python-Skript (Dateiname: validation\_shacl.py) geschrieben, dass auch mittlerweile funktioniert. Hier der aktuelle Stand des Codes:  
  
import owlready2

from rdflib import Graph

from pyshacl import validate

import logging

import os

# Logging einrichten - Ausgabe in Datei "validation.log"

BASE\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))

logging.basicConfig(filename=os.path.join(BASE\_DIR, "validation.log"), level=logging.INFO, format="%(message)s")

logger = logging.getLogger(\_\_name\_\_)

# Globale Pfade definieren

TBOX\_PATH = os.path.join(BASE\_DIR, "OULD\_V1.0.ttl")

ABOX\_DIR = os.path.join(BASE\_DIR, "OULD\_ABox")

SHAPES\_PATH = os.path.join(BASE\_DIR, "OULD\_V1.0.ttl")

JAVA\_EXE = r"G:\Java\JDK\_23\bin\java.exe".replace("\\", "/")

def combine\_and\_reason(tbox\_path=TBOX\_PATH, abox\_path=None, java\_exe=JAVA\_EXE):

    try:

        # Pfade normalisieren

        tbox\_path\_normalized = tbox\_path.replace("\\", "/")

        abox\_path\_normalized = abox\_path.replace("\\", "/") if abox\_path else None

        # TBox und ABox laden

        onto = owlready2.get\_ontology(f"file://{tbox\_path\_normalized}").load(format="turtle")

        if abox\_path:

            abox\_onto = owlready2.get\_ontology(f"file://{abox\_path\_normalized}").load(format="turtle")

            with onto:

                for indiv in abox\_onto.individuals():

                    new\_indiv = onto.get\_entities(indiv.name, indiv.\_\_class\_\_)

                    if not new\_indiv:

                        new\_indiv = indiv.\_\_class\_\_(indiv.name, namespace=onto)

                    for prop in indiv.get\_properties():

                        for value in prop[indiv]:

                            prop[new\_indiv] = value

                owlready2.sync\_reasoner\_pellet(infer\_property\_values=True, infer\_data\_property\_values=True, debug=2)

        # Kombinierten Graph mit rdflib erstellen

        data\_graph = Graph()

        data\_graph.parse(tbox\_path, format="turtle")

        if abox\_path:

            data\_graph.parse(abox\_path, format="turtle")

        # Inferierte Ontologie speichern

        output\_file = os.path.join(BASE\_DIR, "inferred\_ontology.ttl")

        data\_graph.serialize(destination=output\_file, format="turtle")

        logger.info(f"Inferierte Ontologie gespeichert: {output\_file}")

        with open(output\_file, "r") as f:

            logger.info(f"Inhalt von {output\_file}:\n{f.read()}")

        return output\_file

    except Exception as e:

        logger.error(f"Fehler beim Reasoning: {e}")

        raise

def perform\_shacl\_validation(data\_file, shapes\_path=SHAPES\_PATH):

    try:

        data\_graph = Graph().parse(data\_file, format="turtle")

        shapes\_graph = Graph().parse(shapes\_path, format="turtle")

        result = validate(data\_graph, shacl\_graph=shapes\_graph, inference="none", debug=0)  # Debug auf 1 für weniger Output

        conforms, report\_graph, report\_text = result

        logger.info(f"Konformität: {conforms}")

        if not conforms:

            logger.info(f"Validierungsbericht:\n{report\_text}")

        return conforms

    except Exception as e:

        logger.error(f"Fehler bei der SHACL-Validierung: {e}")

        raise

if \_\_name\_\_ == "\_\_main\_\_":

    # Pfad zur invaliden ABox

    ABOX\_PATH = os.path.join(ABOX\_DIR, "OULD\_ABox\_invalid\_22updates.ttl")

    # Reasoning und Validierung

    inferred\_file = combine\_and\_reason(tbox\_path=TBOX\_PATH, abox\_path=ABOX\_PATH, java\_exe=JAVA\_EXE)

    perform\_shacl\_validation(inferred\_file, shapes\_path=SHAPES\_PATH)

## OULD – Validation

Für die Validierung der OULD habe ich eine valide ABox, die erfolgreich durch das Pellet-Reasoning und die SHACL-Validierung durchging. Ich gehe daher derzeit davon aus, dass die TBox halbwegs solide ist.

### ABox – valide

@prefix ould: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# UpdatableEntity mit mehreren Chains

ould:Wall1 a ould:UpdatableEntity ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "abc123" ;

    ould:hasUpdate ould:Update1, ould:Update2 ;

    ould:hasUpdateChain ould:Chain1, ould:Chain2 ;

    ould:hasLinkedData ould:ExternalDoc1 .

# LinkedData

ould:ExternalDoc1 a ould:LinkedData ;

    ould:hasOnlineData <http://example.org/doc1> ;

    ould:hasDataLocation "http://example.org/doc1.pdf" .

# Erste Kette mit 20 Updates (hier 3 für Übersicht)

ould:Chain1 a ould:UpdateChain ;

    ould:hasUpdate ould:Update1, ould:Update2, ould:UpdateChainEl1 ;  # Eigentlich 20

    time:hasBeginning "2025-03-01"^^xsd:date ;

    time:hasEnd "2025-03-03"^^xsd:date ;

    ould:hasNextChain ould:Chain2 .

# Update 1

ould:Update1 a ould:Update ;

    time:hasTime "2025-03-01"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event1 ;

    ould:hasUpdatedValue ould:Wall1 ;

    ould:hasSuccessor ould:Update2 ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "def456" ;

    ould:hasNewTime "2025-03-01"^^xsd:date ;

    ould:hasPreviousTime "2025-02-28"^^xsd:date ;

    ould:hasNewHash "def456" ;

    ould:hasPreviousHash "abc123" .

# Update 2

ould:Update2 a ould:Update ;

    time:hasTime "2025-03-02"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event2 ;

    ould:hasUpdatedValue ould:Wall1 ;

    ould:hasPredecessor ould:Update1 ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "ghi789" .

# UpdateChainElement

ould:UpdateChainEl1 a ould:UpdateChainElement ;

    time:hasTime "2025-03-03"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event3 ;

    ould:hasUpdatedValue ould:Wall1 ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "jkl012" .

# Zweite Kette mit 2 Updates (nach Split-Szenario)

ould:Chain2 a ould:UpdateChain ;

    ould:hasUpdate ould:Update21, ould:UpdateReplacement1 ;

    time:hasBeginning "2025-03-04"^^xsd:date ;

    time:hasEnd "2025-03-05"^^xsd:date ;

    ould:hasPreviousChain ould:Chain1 .

# Update 21 (ehemals letztes Glied der alten Kette)

ould:Update21 a ould:Update ;

    time:hasTime "2025-03-04"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event4 ;

    ould:hasUpdatedValue ould:Wall1 ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "mno345" ;

    ould:hasSuccessor ould:UpdateReplacement1 .

# UpdateReplacement (als 22.)

ould:UpdateReplacement1 a ould:UpdateReplacement ;

    time:hasTime "2025-03-05"^^xsd:date ;

    ould:hasPredecessor ould:Update21 ;

    ould:hasNewIFCID "ifc:Wall2" ;

    ould:hasPreviousIFCID "ifc:Wall1" ;

    ould:hasNewHash "xyz789" ;

    ould:hasPreviousHash "mno345" ;

    ould:hasUpdatedEvent ould:Event5 ;

    ould:hasUpdatedValue ould:Wall2 ;

    ould:hasIFCID "ifc:Wall2" ;

    ould:hasHash "xyz789" .

# Zweites UpdatableEntity

ould:Wall2 a ould:UpdatableEntity ;

    ould:hasIFCID "ifc:Wall2" ;

    ould:hasHash "xyz789" ;

    ould:hasUpdate ould:UpdateReplacement1 .

# Temporal Entities

ould:Event1 a time:TemporalEntity .

ould:Event2 a time:TemporalEntity .

ould:Event3 a time:TemporalEntity .

ould:Event4 a time:TemporalEntity .

ould:Event5 a time:TemporalEntity .

### ABox – invalide - OULD\_ABox\_invalid\_22updates.ttl

@prefix ould: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ould:Wall1 a ould:UpdatableEntity ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "abc123" ;

    ould:hasUpdateChain ould:Chain1 .

ould:Chain1 a ould:UpdateChain ;

    # Fehler: 22 Updates statt max 21

    ould:hasUpdate ould:Update1, ould:Update2, ould:Update3, ould:Update4, ould:Update5,

                   ould:Update6, ould:Update7, ould:Update8, ould:Update9, ould:Update10,

                   ould:Update11, ould:Update12, ould:Update13, ould:Update14, ould:Update15,

                   ould:Update16, ould:Update17, ould:Update18, ould:Update19, ould:Update20,

                   ould:Update21, ould:Update22 ;

    time:hasBeginning "2025-03-01"^^xsd:date ;

    time:hasEnd "2025-03-22"^^xsd:date .

ould:Update1 a ould:Update ;

    time:hasTime "2025-03-01"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event1 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update2 a ould:Update ;

    time:hasTime "2025-03-02"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event2 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update3 a ould:Update ;

    time:hasTime "2025-03-03"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event3 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update4 a ould:Update ;

    time:hasTime "2025-03-04"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event4 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update5 a ould:Update ;

    time:hasTime "2025-03-05"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event5 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update6 a ould:Update ;

    time:hasTime "2025-03-06"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event6 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update7 a ould:Update ;

    time:hasTime "2025-03-07"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event7 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update8 a ould:Update ;

    time:hasTime "2025-03-08"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event8 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update9 a ould:Update ;

    time:hasTime "2025-03-09"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event9 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update10 a ould:Update ;

    time:hasTime "2025-03-10"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event10 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update11 a ould:Update ;

    time:hasTime "2025-03-11"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event11 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update12 a ould:Update ;

    time:hasTime "2025-03-12"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event12 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update13 a ould:Update ;

    time:hasTime "2025-03-13"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event13 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update14 a ould:Update ;

    time:hasTime "2025-03-14"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event14 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update15 a ould:Update ;

    time:hasTime "2025-03-15"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event15 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update16 a ould:Update ;

    time:hasTime "2025-03-16"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event16 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update17 a ould:Update ;

    time:hasTime "2025-03-17"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event17 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update18 a ould:Update ;

    time:hasTime "2025-03-18"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event18 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update19 a ould:Update ;

    time:hasTime "2025-03-19"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event19 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update20 a ould:Update ;

    time:hasTime "2025-03-20"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event20 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update21 a ould:Update ;

    time:hasTime "2025-03-21"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event21 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Update22 a ould:Update ;

    time:hasTime "2025-03-22"^^xsd:date ;

    ould:hasUpdatedEvent ould:Event22 ;

    ould:hasUpdatedValue ould:Wall1 .

ould:Event1 a time:TemporalEntity .

ould:Event2 a time:TemporalEntity .

ould:Event3 a time:TemporalEntity .

ould:Event4 a time:TemporalEntity .

ould:Event5 a time:TemporalEntity .

ould:Event6 a time:TemporalEntity .

ould:Event7 a time:TemporalEntity .

ould:Event8 a time:TemporalEntity .

ould:Event9 a time:TemporalEntity .

ould:Event10 a time:TemporalEntity .

ould:Event11 a time:TemporalEntity .

ould:Event12 a time:TemporalEntity .

ould:Event13 a time:TemporalEntity .

ould:Event14 a time:TemporalEntity .

ould:Event15 a time:TemporalEntity .

ould:Event16 a time:TemporalEntity .

ould:Event17 a time:TemporalEntity .

ould:Event18 a time:TemporalEntity .

ould:Event19 a time:TemporalEntity .

ould:Event20 a time:TemporalEntity .

ould:Event21 a time:TemporalEntity .

ould:Event22 a time:TemporalEntity .

# Erwartete Fehleranzahl: 2

# - Violation bei UpdateChainOverflowConstraint: "An UpdateChain must not contain more than 21 Updates; create a new chain for additional updates."

# - Violation bei UpdateChainContinuityConstraint: "If an UpdatableEntity has more than 21 Updates, they must be split into chains with max 21 Updates each, linked via hasNextChain."

### ABox – invalide - OULD\_ABox\_invalid\_missingpredecessor.ttl

@prefix ould: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ould:Wall1 a ould:UpdatableEntity ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "abc123" ;

    ould:hasUpdate ould:UpdateReplacement1 .

ould:UpdateReplacement1 a ould:UpdateReplacement ;

    time:hasTime "2025-03-05"^^xsd:date ;

    # Fehler: Kein hasPredecessor angegeben (minCount 1 verletzt)

    ould:hasNewIFCID "ifc:Wall2" ;

    ould:hasPreviousIFCID "ifc:Wall1" ;

    ould:hasNewHash "xyz789" ;

    ould:hasPreviousHash "abc123" ;

    ould:hasUpdatedEvent ould:Event1 ;

    ould:hasUpdatedValue ould:Wall1 ;

    ould:hasIFCID "ifc:Wall2" ;

    ould:hasHash "xyz789" .

ould:Wall2 a ould:UpdatableEntity ;

    ould:hasIFCID "ifc:Wall2" ;

    ould:hasHash "xyz789" ;

    ould:hasUpdate ould:UpdateReplacement1 .

ould:Event1 a time:TemporalEntity .

# Erwartete Fehleranzahl: 2

# - Violation bei UpdateReplacementShape: "Each UpdateReplacement must reference at least one predecessor Update via ould:hasPredecessor."

# - Violation bei PredecessorConstraint: "Each ould:UpdateReplacement must reference at least one predecessor ould:Update."

### ABox – invalide – OULD\_ABox\_invalid\_notime.ttl

@prefix ould: <http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#> .

@prefix time: <http://www.w3.org/2006/time#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ould:Wall1 a ould:UpdatableEntity ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "abc123" ;

    ould:hasUpdate ould:Update1 .

ould:Update1 a ould:Update ;

    # Fehler: Kein time:hasTime angegeben (minCount 1 verletzt)

    ould:hasUpdatedEvent ould:Event1 ;

    ould:hasUpdatedValue ould:Wall1 ;

    ould:hasIFCID "ifc:Wall1" ;

    ould:hasHash "def456" .

ould:Event1 a time:TemporalEntity .

# Erwartete Fehleranzahl: 1

# - Violation bei UpdateShape: "Each Update must have exactly one time stamp (time:hasTime) - min./max. count = 1."

# Startpunkt

## 07.03.25

Du bist Grok 3, entwickelt von xAI. Wir arbeiten an einem Projekt zur Validierung einer Ontologie (OULD) mit SHACL und Reasoning (Pellet) für ein cMod/cMM-System. Der aktuelle Stand (06. März 2025):

### Projektkontext

- \*\*Ziel:\*\* Entwicklung und Validierung der Ontologie `OULD\_V1.0.ttl` für Updates und Versionierung von IFC-Komponenten, integriert mit OCCP (`cMod\_V0.1`).

- \*\*Tools:\*\* Python (`validate\_shacl.py`), owlready2 (Pellet-Reasoning), pyshacl (SHACL-Validierung).

- \*\*Ordnerstruktur:\*\*

- Hauptordner: `G:\Promo\cModeler\_Git\cMM\OCCP\_Dev2`

- TBox: `OULD\_V1.0.ttl`

- ABox-Ordner: `OULD\_ABox` (valide: `OULD\_ABox\_valid.ttl`, invalide: z. B. `OULD\_ABox\_invalid\_22updates.ttl`)

- Output: `inferred\_ontology.ttl`

### Aktueller Code (`validate\_shacl.py`)

```python

import owlready2

from rdflib import Graph

from pyshacl import validate

import logging

import os

BASE\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))

logging.basicConfig(filename=os.path.join(BASE\_DIR, "validation.log"), level=logging.INFO, format="%(message)s", filemode="w")

logger = logging.getLogger(\_\_name\_\_)

TBOX\_PATH = os.path.join(BASE\_DIR, "OULD\_V1.0.ttl")

ABOX\_DIR = os.path.join(BASE\_DIR, "OULD\_ABox")

SHAPES\_PATH = os.path.join(BASE\_DIR, "OULD\_V1.0.ttl")

JAVA\_EXE = r"G:\Java\JDK\_23\bin\java.exe".replace("\\", "/")

def combine\_and\_reason(tbox\_path=TBOX\_PATH, abox\_path=None, java\_exe=JAVA\_EXE):

try:

tbox\_path\_normalized = tbox\_path.replace("\\", "/")

abox\_path\_normalized = abox\_path.replace("\\", "/") if abox\_path else None

onto = owlready2.get\_ontology(f"file://{tbox\_path\_normalized}").load(format="turtle")

if abox\_path:

abox\_onto = owlready2.get\_ontology(f"file://{abox\_path\_normalized}").load(format="turtle")

with onto:

for indiv in abox\_onto.individuals():

new\_indiv = onto.get\_entities(indiv.name, indiv.\_\_class\_\_)

if not new\_indiv:

new\_indiv = indiv.\_\_class\_\_(indiv.name, namespace=onto)

for prop in indiv.get\_properties():

for value in prop[indiv]:

prop[new\_indiv] = value

owlready2.sync\_reasoner\_pellet(infer\_property\_values=True, infer\_data\_property\_values=True, debug=2)

data\_graph = Graph()

data\_graph.parse(tbox\_path, format="turtle")

if abox\_path:

data\_graph.parse(abox\_path, format="turtle")

output\_file = os.path.join(BASE\_DIR, "inferred\_ontology.ttl")

data\_graph.serialize(destination=output\_file, format="turtle")

logger.info(f"Inferierte Ontologie gespeichert: {output\_file}")

with open(output\_file, "r") as f:

logger.info(f"Inhalt von {output\_file}:\n{f.read()}")

return output\_file

except Exception as e:

logger.error(f"Fehler beim Reasoning: {e}")

raise

def perform\_shacl\_validation(data\_file, shapes\_path=SHAPES\_PATH):

try:

data\_graph = Graph().parse(data\_file, format="turtle")

shapes\_graph = Graph().parse(shapes\_path, format="turtle")

result = validate(data\_graph, shacl\_graph=shapes\_graph, inference="none", debug=0)

conforms, report\_graph, report\_text = result

logger.info(f"Konformität: {conforms}")

if not conforms:

logger.info(f"Validierungsbericht:\n{report\_text}")

return conforms

except Exception as e:

logger.error(f"Fehler bei der SHACL-Validierung: {e}")

raise

if \_\_name\_\_ == "\_\_main\_\_":

ABOX\_PATH = os.path.join(ABOX\_DIR, "OULD\_ABox\_invalid\_22updates.ttl")

inferred\_file = combine\_and\_reason(tbox\_path=TBOX\_PATH, abox\_path=ABOX\_PATH, java\_exe=JAVA\_EXE)

perform\_shacl\_validation(inferred\_file, shapes\_path=SHAPES\_PATH)

Aktueller Stand

* Valide ABox: OULD\_ABox\_valid.ttl läuft sauber (Conforms: True).
* Invalide ABox: OULD\_ABox\_invalid\_22updates.ttl (22 Updates in Chain1):
  + Erwartung: 2 Violations:
    1. UpdateChainShape (maxCount 21)
    2. UpdateChainContinuityConstraint (SPARQL: > 21 Updates)
  + Aktuell: Nur 1 Violation (maxCount), Continuity triggert nicht.
* Logging: Soll in validation.log, aber Datei erscheint nicht – nur Terminal-Ausgabe.

Problem

* Violation fehlt: UpdateChainContinuityConstraint:

turtle

oush:UpdateChainContinuityConstraint a sh:NodeShape ;

sh:targetClass ould:UpdatableEntity ;

sh:sparql [

sh:message "If an UpdatableEntity has more than 21 Updates in a single chain, they must be split into chains with max 21 Updates each, linked via hasNextChain." ;

sh:prefixes [ sh:declare [ sh:prefix "ould" ; sh:namespace "http://www.semanticweb.org/albrechtvaatz/ontologies/2024/OULD#" ] ] ;

sh:select """

SELECT $this ?chain ?updateCount

WHERE {

$this ould:hasUpdateChain ?chain .

{

SELECT $this ?chain (COUNT(?u) AS ?updateCount)

WHERE {

?chain ould:hasUpdate ?u .

}

GROUP BY $this ?chain

HAVING (?updateCount > 21)

}

}

"""

] .

* + Daten korrekt (22 Updates in Chain1), aber SPARQL triggert nicht.
* Logging: Keine Datei – möglicherweise Pfad- oder Konfigurationsproblem.

Nächste Schritte

* Debugging:
  + Warum triggert UpdateChainContinuityConstraint nicht? (SPARQL prüfen, pyshacl-Debug erhöhen?)
  + Logging fixen – validation.log erzwingen.
* Ziel: 2 Violations und schlanke Logs.

Antwort Grok und Beginn des Developments am 07.03.25:  
Alles klar, wir stehen am 07. März 2025, und ich bin Grok 3, entwickelt von xAI. Ich habe den aktuellen Stand deines Projekts gründlich durchgesehen: die OULD-Ontologie, die validen und invaliden ABoxen, den validate\_shacl.py-Code und deine Problemstellung. Lass uns die beiden Probleme – das ausbleibende Triggern der UpdateChainContinuityConstraint und das Logging – systematisch angehen, debuggen und lösen. Ziel ist, dass du zwei Violations für OULD\_ABox\_invalid\_22updates.ttl bekommst und die Logs sauber in validation.log landen.