Model Development

3

Alexander Kulesza, Axel Loewe, Andrea Stenti, Chiara Nicolò, Enrique Morales-Orcajo, Eulalie Courcelles, Fianne Sips, Francesco Pappalardo, Giulia Russo, Marc Horner, Marco Viceconti, Martha De Cunha Maluf-Burgman, Raphaëlle Lesage, and Steve Kreuzer

3.1 A Risk-Based Paradigm of Model Development as a Function of Its Context of Use

Good Simulation Practice implies that a computational model considered for a simulation task has also been developed according to good practice. In this Chapter, an attempt is made to summarise and synthesise good practices in computational model development. A high level of abstraction is needed when considering numerous different model types (a recent report of the US Food and Drug Administration (FDA)[1] mentions 39 different modelling classes). Therefore, this Chapter focuses on model development and implementation as a process rather than concrete model-type specific recommendations. Generic model definition and design recommendations are addressed in Sect. 3.2.2.

[1] https://www.fda.gov/media/163156/download.

A. Kulesza (✉) · E. Courcelles
Novadiscovery, Lyon, France
e-mail: alexander.kulesza@novadiscovery.com

E. Courcelles
e-mail: eulalie.courcelles@novadiscovery.com

A. Loewe
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mail: axel.loewe@kit.edu

A. Stenti
ULB—Université Libre de Bruxelles, Bruxelles, Belgium
e-mail: andrea.stenti@ulb.be

C. Nicolò
InSilicoTrials Technologies, Trieste, Italy
e-mail: chiara.nicolo@insilicotrials.com

M. Viceconti and L. Emili (eds.), *Toward Good Simulation Practice*, Synthesis Lectures on Biomedical Engineering, https://doi.org/10.1007/978-3-031-48284-7_3

Whether one develops a predictive model from scratch or from existing libraries and solvers, computational model development shares many commonalities with software development:

(a) Models transform user inputs into outputs.
(b) Models can be developed as standalone units or as part of larger systems/platforms.
(c) A model's life cycle is similar to that of software.
(d) The concrete implementation of a predictive model is often part of a software.

Thus, it is reasonable to explore existing standards for Software Life Cycle (SLC) management (systems and software engineering) as a starting point for good practices in

E. Morales-Orcajo
Ambu, Augsburg, Germany
e-mail: enmo@ambu.com

F. Sips
InSilicoTrials Technologies, 's-Hertogenbosch, Netherlands
e-mail: fianne.sips@insilicotrials.com

F. Pappalardo · G. Russo
University of Catania, Catania, Italy
e-mail: francesco.pappalardo@unict.it

G. Russo
e-mail: giulia.russo@unict.it

M. Horner
Ansys Inc., Evanston, USA
e-mail: marc.horner@ansys.com

M. Viceconti
Alma Mater Studiorum—University of Bologna, Bologna, Italy
e-mail: marco.viceconti@unibo.it

M. De Cunha Maluf-Burgman
Edwards Lifesciences, Den Haag, Netherlands
e-mail: martha_cunhamaluf-burgman@edwards.com

R. Lesage
VPH Institute, Leuven, Belgium
e-mail: manager@vph-institute.org

S. Kreuzer
Exponent, Boston, USA
e-mail: skreuzer@exponent.com

model development, defined according to widely agreed-upon "best ways of doing" it[2]—relevant for application and uptake in mission-critical and highly regulated environments. Many different programming languages and software development paradigms exist and can be used for developing computational models under consideration of process-level software development good practice—agnostic of procedural and content-related details. We will therefore leverage the similarity between computational models and software to map good software development practices onto the former.

Model developers must acknowledge that their "product" may operate under a "regulated" environment and that regulators will perform a benefit-risk assessment. Any regulatory effort in a mission-critical domain faces the challenge of balancing the need for the lowest possible level of risk for the patient and the economic viability of product development, without which no product could be brought to market. Risk assessment, clinical evaluation, and validation revolve around the "intended purpose" defined by the Medical Device Regulation (MDR). There is a debate around how risk management should be implemented in the development of a device: following the concept of risk "As Low As Reasonably Practicable" (ALARP) as proposed by ISO 14971:2019 or risk reduced "As Far As Possible" (AFAP) as requested by the new EU regulations.[3] Concrete regulatory recommendations on transferring these concepts to computational models do not yet exist, except for a FDA guidance document.[4] We, therefore, adopt a risk-based approach for the development of computational models, where the level of scrutiny (in terms of model life cycle management) is proportional to the assessed risk that a predictive model can pose (e.g., for patients) according to pre-defined Context(s) of Use (CoU).

Thus, this Chapter focuses on applying a risk-based approach (Fig. 3.1) for the model and simulation software development process, where good practice establishes a minimal set of process-level requirements for all models (even low-risk ones). At the same time, it requires more comprehensive compliance with relevant industry standards for medium to high-risk scenarios, critical applications, and models with substantial impact on regulatory decision-making. It is also important to note that this chapter focuses on developing the modelling and simulation software platform. In contrast, Chap. 4 focuses on the result of this process, which is the actual implementation (i.e., the model) and its credibility assessment.

In the following, we first highlight some industry standards and a potential mapping to elements of model development. We then iterate through the stages of a life cycle model relevant to model development. In each section, we adopt a viewpoint from low and high model risk to derive two (example) levels of alignment with the cited industry

[2] "ISO standards are internationally agreed by experts." https://www.iso.org/standards.html. Accessed 19 Sept. 2021.

[3] https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32017R0745.

[4] The final FDA guidance is now available (November) https://www.fda.gov/regulatory-information/search-fda-guidance-documents/assessing-credibility-computational-modeling-and-simulation-medical-device-submissions

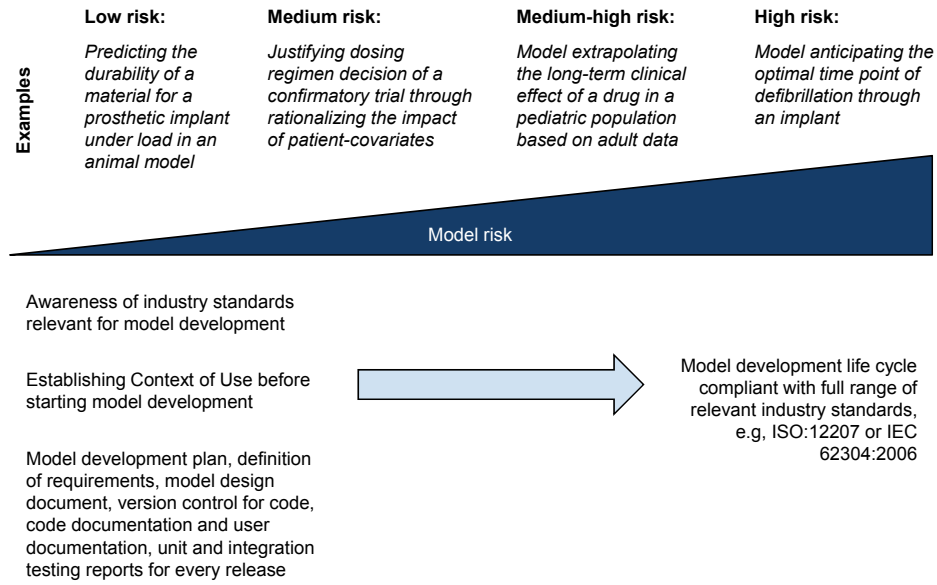

Fig. 3.1 A risk-based approach for model development planning (simplified; for a complete model risk assessment, see ASME VV-40:2018). While a lean plan can suffice for low-risk CoUs, higher-risk projects must gradually consider the full range of relevant industry standards detailed in Sect. 3.2. Note that model risk needs to be anticipated by the developer during development.

standards and derive good simulation practice recommendations for model development (Tables 3.2, 3.3, 3.4, 3.5 and 3.6).

This Chapter's scope is limited to model development practices and does not consider model use and validation aspects, which are covered in Chap. 4.

3.2 SLC Industry Standards and Relevance for Model Development

The previous section introduced a risk-based approach for model development, implementing different levels of compliance with industry standards. As no industry standard for computational models yet exists, the next best option is to adopt and apply standards and best practices from related areas similar to computational model development. Of particular interest to the development of predictive models is the great body of process-level knowledge and recommendations available for software development—not only because of the analogy between software and model development but also because the developed model and the software in which it is implemented are often intertwined.

Two software development standards are relevant to the model development process: ISO/IEC/IEEE:12207[5] and ISO/IEC:62304.[6] The former applies to every software package or system, whereas the latter is specific to medical device software. Other relevant best practice documents include the National Aeronautics and Space Administration (NASA) handbook on model development (NASA-STD-7009A[7] and NASA-HDBK-7009A[8]). A full mapping of all the model development activities, with processes from all potentially applicable industry standards, is out of this Chapter's scope. Instead, we intend to highlight the opportunities where an explicit consideration of industry standards supports overall quality, especially in critical applications.

- **The ISO/IEC/IEEE 12207:2017 standard "Software life cycle processes"** covers different process groups, including (I) Agreement, (II) Organisational project-enabling processes, (III) Technical management processes, and (IV) Technical processes. Process groups III and IV are most relevant for a given model development project in a given organisational structure (for definitions of a process, see, e.g., ISO/IEC/IEEE 24774:2021). We stress that management and technical processes are orthogonal.
  - **Technical management processes** are concerned with managing and applying the resources and assets allocated by the organisation's management and applying them. Technical management comprises (a) Project planning, (b) Project assessment and control process, (c) Decision management processes, (d) Risk management processes, (e) Configuration management, (f) Information management, (g) Measurement, and (h) Quality assurance processes.
  - **Technical processes** are concerned with technical actions throughout the life cycle. Technical processes transform the needs of stakeholders into a product or service. Technical processes include (a) Mission analysis, (b) Stakeholder needs and requirements definition, (c) Systems/software requirements definition, (d) Architecture definition, (e) Design definition, (f) System analysis, (g) Implementation, (h) Integration, (i) Verification, (j) Transition, (k) Validation, (l) Operation, (m) Maintenance and (n) Disposal, of which (a–h) may be classified as "development" and are covered in detail in this Chapter while (i–k) are more relevant for model validation (see Chap. 4). Processes (l–n) need to be considered to qualify for sustained use or regulatory approval, where maintenance is often challenging in a research setting (Anzt et al., 2021).
  - ISO/IEC/IEEE 15288:2015 establishes a common framework of process descriptions for the life cycle of systems and is often used in conjunction with ISO/IEC/IEEE 12207:2017. Additionally, ISO/IEC/IEEE 24748–3 describes the application of ISO/IEC/IEEE 12207:2017.

[5] https://www.iso.org/standard/63712.html.
[6] https://www.iso.org/standard/38421.html.
[7] https://standards.nasa.gov/standard/NASA/NASA-STD-7009.
[8] https://standards.nasa.gov/standard/nasa/nasa-hdbk-7009.

- **The standard ISO/IEC 62304:2006 "medical device software—software life cycle (SLC) processes"** may be more readily adopted as it targets regulated environments in healthcare and is recognised by both the European Union as a Harmonised Standard and the United States FDA as a Recognized Consensus Standard. The structure of this standard is similar to ISO/IEC/IEEE 12207 (see overlap indicated in Table 3.1). In fact, modern development platforms that combine the ability to develop, secure, and operate software, can be designed to establish compliance (see here[9] for the case of ISO/IEC 62304).

These ISO/IEC standards do not prescribe any particular life cycle model, acknowledging that software development processes should be oriented towards the project objectives. Instead, they define a set of life cycle processes, which can be used to define the SLC. Depending on the SLC model used, the development phases may be classified as follows:

– **Analysis and Requirements**: derive requirements from the mission analysis, the user perspective and the (integrated) system viewpoint as a function of the real-world system and potential application of M&S.
– **Design**: collection of information and definition of concepts to include in the proposed model; the iterative process of creating the detailed, verifiable, and validated model specification and simulations for the intended use.
– **Implementation and Integration**: realisation of the technical implementation of the model design in line with the requirements, specifications, and intended use.
– **Testing**: checking to determine if the model meets all requirements and operational intentions.
– **Maintenance**: release of the software, archiving of artefacts, life cycle management.

Orthogonal to the technical processes, adherence to management processes can be beneficial or required. Also, adherence to industry standards rarely concerns one portion of the set of business processes but more likely impacts a large set of operations. Other standards covering quality management/assurance also apply, such as ISO 13485 (quality management system for designing and manufacturing medical devices) or, even more generally, ISO 9001. Also, consideration of a service management system specified by ISO/IEC 20000-1, an IT asset management system specified by ISO/IEC 19770 (all parts), and an information security management system specified by ISO/IEC 27000 may be relevant.

[9] https://about.gitlab.com/solutions/iec-62304/.

Table 3.1 Summary of relationships between the phases outlined in this document and most relevant phases and processes in selected industry standards and best practice documents on model development planning

SLC phase	ISO IEC 12207	ISO 62304	NASA-HDBK-7009A
Analysis	Mission analysis	Software development planning	Model initiation
Requirements	Stakeholder needs and requirements definition	Software requirements analysis	
	Systems/software requirements definition		
Design	Architecture definition	Software architectural design	Model concept development
	Design definition	Software detailed design	Model design
	System analysis		
Implementation	Implementation	Software unit implementation and verification	Model construction
	Integration	Software integration and integration testing	
Testing	Verification (developer and end user)	Software system testing	Model testing and release
	Transition		
	Validation (Developer and end user)		
Maintenance	Operation (end user)	Software release	Model use (end user)
	Maintenance		Model and Analysis Archiving (developer and end user)
	Disposal		

3.2.1 Analysis and Requirements

The software development life cycle is initiated by a planning phase where the overall mission, problem, and context are analysed, and the actual requirements are defined. Subsequently, a plan defines how the new development will fulfil the mission. In many life cycle models, this initial phase is called Elicitation. Depending on the model, a development plan that is either more project-oriented or more technically oriented can be the better choice. Table 3.2 lists good model development practices by establishing, considering and documenting a model development plan and the requirements definitions

Table 3.2 Good practices for the analysis and requirements phase of model development on the low and high-risk ends of the risk spectrum (see Fig. 3.1, left and right, respectively)

	Low Risk	**High Risk**
Requirement definitions (design prerequisite)	Mission requirements (CoU) Risks User requirements System requirements (if any)	Concept of Operations (ConOps) document System Requirements Document (SRD) Requirements Traceability Matrix (RTM) See IEEE/ISO/IEC 29148-2018
Model development plan (project management)	**Model development plan** similar to a software management plan, see e.g. (The Software Sustainability Institute, 2018) Reference Materials (including knowledge and data sources) Development and life cycle planning	Detailed **model development plan** similar to a software development plan. See ISO 62304-2006 and Rust et al. (2016)

document(s) for both low and high model risk scenarios (Fig. 3.1 left and right, respectively). Generally, we regard the high-risk recommendation as the gold standard, while simplified processes and documentation can be acceptable for low-risk situations.

From a model developer viewpoint, this phase in the SLC overlaps with the definition of a CoU in ASME VV-40:2018 (see Table 3.1). The model developer must anticipate the required CoUs that the model will aim for (the CoUs anticipated by the developer might be captured in a Concept of Operations document and use cases). CoU formulation should be embedded in high-risk contexts with requirements definitions aligned with industry standards from related disciplines until specific ones become available.

Planning for the project is often captured in a Project Management Plan. ISO/IEC/IEEE 16326 provides more detail on project planning. The project planning process aims to produce and coordinate effective, workable plans. This process determines the scope of the project management and technical activities, identifies process outputs, tasks, and deliverables, and establishes schedules for conducting tasks, including achievement criteria and required resources to accomplish tasks. Project planning is an ongoing process throughout a project with regular plan revisions. Technical planning for a software system is often captured in a Systems Engineering Management Plan, a Software Engineering Management Plan, or a Software Development Plan (SDP). ISO/IEC/IEEE 24748-5 provides more detail on software engineering technical management planning and includes an annotated outline for an SDP. Notably, ISO 62304-2006 and Rust et al. (2016) suggest an SDP structure commensurate with regulated environments.

Good practice of model development should establish such a development plan (roughly for "low risk" or as precisely as possible for "high risk"; Table 3.2) considering the related guidance on software development.

Another important stage in the initial development life cycle phase of Elicitation is the definition of requirements. Requirements can come from many different sources, for example, user needs, functionality, performance, risk, regulatory, processes, or marketing. As stipulated in Table 3.2 (top row), both "low-risk" and "high-risk" models should document the requirements for their development. Explicitly adhering to industry standards might not be required for low-risk models.

The requirements definition process captures and transforms stakeholder needs into "well-formed requirements" (suitable as inputs for subsequent model development procedures). A "well-formed requirement" shall possess the following attributes: necessary, appropriate, unambiguous, complete, singular, feasible, verifiable, correct, and conforming. IEEE/ISO/IEC 29148-2018 provides more detail on requirements engineering and requirement processes. The user and system requirements should be likewise defined with the formulated CoUs (equivalent to mission requirements). The entire set of requirements enables a common understanding between stakeholders and provides a reference for verification. They must also be validated against real-world needs and be feasible to implement and check (potentially formulated as part of a System Requirements Document (SRD) or Requirements Traceability Matrix (RTM) in high-risk contexts). This enables users to practically judge whether usage scenarios are within the intended CoUs versus ones that might be technically possible but outside of the CoUs.

3.2.2 Design

The key prerequisite for model design is the definition of the CoU(s) during the requirements specification. In the "design specification," these CoU(s) must be translated into the architecture and component design of the actual model. As introduced in Chap. 2, the formulation of the model in terms of fundamental mathematical equations and parameters is crucial. On the one hand, the model needs to be complex enough to fulfil its CoU(s). On the other hand, unique parameter identifiability and parameter uncertainty (either on a population level during calibration or in a given subject during personalisation depending on the CoU) (Galappaththige et al., 2022; Parvinian et al., 2019), numerical accuracy and required computational effort as well as the options for verification and validation (Pathmanathan & Gray, 2014; Pathmanathan et al., 2017) need to be considered. Following the law of parsimony, one should aim for the simplest model that can support the intended CoU. The decision-making process and limitations of this choice must be explicitly documented (Erdemir et al., 2019). Table 3.3 lists specific aspects to be considered during the Design phase for low and high-risk applications.

Table 3.3 Good practices for the design phase of model development on the low and high-risk ends of the risk spectrum

	Low Risk	**High Risk**
Model design	Simple **model design document** **Definition of a conceptual model** Which modelling approach is suitable for the CoU? What level of precision is required? Document the limitations of the model **Definition of the architecture design** Focus on functionality, covered hypotheses and phenomena **Description of the detailed design** Considering model-type specific recommendations Focus on expected sensitivity, identifiability, **User interfaces** Human-machine interfaces User experience Dialog design Presentation of information **More detailed operational scenarios and use cases** For example, as a simulation plan or protocol (Developed together with Sponsor (see Chapter 10))	Comprehensive **model design document** (including the elements from low risk, compliant with relevant design documentation standards see for example[a]) Additionally: **Definition of the architecture design** Additionally (if relevant) describe design decisions related to performance compatibility transferability usability adaptability reliability security maintainability data privacy **User interfaces** Describe also measures to avoid (user) errors, avoid misinterpretation, to increase use efficiency (ergonomics) and user satisfaction Consider also Usability Engineering File (ISO 14971, IEC 62366-1) **More detailed operational scenarios and use cases** Document decision process according to standards (e.g., ISO 13485, IEC 62304)
Model validation plan	See Chapter 4 and FDA guidance[b]	

[a] https://ntrs.nasa.gov/api/citations/20160011412/downloads/20160011412.pdf

[b] "General Principles of Software Validation; Final Guidance for Industry and FDA Staff", https://www.fda.gov/media/73141/download

Bäker (2018) gave recommendations for computational simulations relevant to operational scenarios and use cases (to be specified by the developer) as well as the simulations to be performed by the end user (CoUs). The initial Concept of Operations document (see the previous section) has to be updated to consider any limitations by the chosen architecture and further operational details.

As evident from the "low-risk" scenario, comprehensive documentation for the model design is needed, irrespective of the risk. The minimum requirement for all models is thus to justify alignment of the modelling concept (and potentially data flow) with the anticipated CoU and to document fundamental design choices (ODE, PDE, agent-based model, etc., the granularity) and associated limitations. In all cases, the architecture design (phenomena and hypotheses, composition, input–output transformations) and the detailed design (geometry, equations, parameters, initial/boundary conditions) must be documented and justified. This potentially includes the workflow with which unknown parameters are estimated from target data or other workflows to produce tailored versions of the model (mapping onto a geometry, treatments, outputs of other simulations etc.). For many fields, model-type-specific recommendations exist and should be considered, e.g., for pharmacological modelling (Byon et al., 2013; Cucurull-Sanchez et al., 2019; Jean et al., 2021; Ke et al., 2016; Overgaard et al., 2015; Zhao et al., 2012). A comprehensive list of all domains is beyond the scope of this Chapter.

Considering relevant external conditions, the decision to use a specific model architecture should be based on the defined requirements (see the previous section). The essential properties of the architecture are often defined by the required internal and external interfaces to be implemented. The design should use established community standards regarding data formats and application programming interfaces (APIs) whenever possible. One should generally favour simple architectures following the established best practices in software engineering, such as information hiding, loose coupling, high cohesion, separation of concerns and hierarchical decomposition.

Design decisions are documented in the software management plan, possibly created during the Elicitation phase. Model design documentation should be similar across all model risks, even though the extent and form may differ. Good practice in defining and describing the architecture and system design is using diagrams and (standard) graphical notations, such as the Unified Modelling Language (UML)[10], to help communicate with stakeholders, explore potential designs, validate the software architecture, and document decisions. Alongside the input and output data descriptions, detailing how these data enter and leave the system (i.e., interfaces) is also mandatory. Other relevant elements of the documentation (for example, a User Manual) can only be generated during/after implementation (see Sect. 3.2.3).

[10] Unified Modelling Language v2.5.1 (2017) by the OMG standardisation group: https://www.omg.org/spec/UML/.

Notice that the design of the model should anticipate verification and validation activities (either by the developer and/or even the user) to be compatible with them. Therefore, a validation plan should be initiated during the design phase (see Chap. 4).

As can be seen on the right side of Table 3.3, full compliance with industry standards relevant for models can necessitate exhaustive documentation of potential pitfalls, errors, practical and life cycle aspects, and may need to follow certain forms.

3.2.3 Implementation and Integration

The purpose of the implementation process is to realise a specified system element. This process transforms requirements, architecture, and design (including interfaces) into actions that create a system element according to the practices of the selected implementation technology. This process results in a system element that satisfies specified system requirements, architecture, and design.

The integration process aims to synthesise a set of system elements into a functional system (product or service) that satisfies system/software requirements, architecture, and design. This process assembles the implemented system elements. Previously defined interfaces are activated to enable the interoperability of the system elements as intended.

During implementation and integration (Table 3.4), established software engineering best practices should be applied (Anzt et al., 2021; Rust et al., 2016). Fundamental requirements specify using a version control system, such as git, and software documentation in the form of the source code and a user manual. Requirements and issues should be tracked and linked using an adequate infrastructure. Specific versions of the implemented model must be assigned unique identifiers (e.g., build number and date). In the case of software as a medical device, persistent unique identifiers are required. Release software versions should be archived with relevant artefacts like documentation, test reports, etc. and should consider the FAIR principles for research software where applicable (Chue Hong et al., 2021).

Table 3.4 Good practices for the implementation and integration phase of model development on the low-risk and high-risk ends of the risk spectrum

	Low Risk	**High Risk**
Versioned model	Storage of code versions in **version control** systems, e.g., git. **Documentation** of all model versions (both in the form of the source code and a user manual).	Version control, e.g., in line with ISO 12207, through a **Configuration Management process** for the selection of configuration items to be integrated.

As discussed in the next section, automated tests can help increase the efficiency of implementation efforts.

The more regulated the environment and the higher the risk of the CoU, the more important it becomes to strictly standardise these processes. Such standards can, for example, include code style guidelines and conventions.

3.2.4 Testing

Testing serves the purpose of checking whether the model was implemented correctly. It answers the question "Did we build the model right?" as opposed to validation, which addresses "Did we build the right model?" Both aspects are covered in detail in Chap. 4, so we only point out a few specific issues from a model developer's perspective in Table 3.5.

Various forms of testing can be applied during testing, including:

- Regression tests, which run the model with specified input parameters and compare it against previously computed results.
- Simplified test cases with analytical solutions enable an evaluation of the quality of the solution.
- Performance tests can help obtain efficient code but are not the focus of the GSP.

Regardless of the approach, testing should be automated as part of continuous integration pipelines. This will increase adoption and adherence by minimising developer efforts to maintain compliance.

Table 3.5 Good practices for the testing phase of model development on the low-risk and high-risk ends of the risk spectrum

	Low risk	High risk
Tests	Automatic tests during development: – Integration tests – System tests/regression tests	Additional automatic tests throughout the life cycle: – Unit tests – Consider static analysis – Maximize unit testing code coverage
User feedback	Optional	Essential

3.2.5 Maintenance

The need for maintenance can arise from multiple causes other than model bugs, such as version updates of the model solver, functionality, material laws or any of its constituent parts, changes in external dependencies (e.g., software libraries, compilers) or changes in the regulatory framework, requiring re-evaluation of specific credibility factors. Lehman's laws of software evolution postulate that software must continuously evolve to remain useful (Lehman, 1980). This section focuses on the maintenance of the model itself (developer's perspective) rather than the maintenance of specific simulations (user's perspective). It assumes that a model is developed for several uses by different users.

The model development plan should document the maintenance strategy (ISO/IEC/IEEE 12207:2017). Medical device regulation (MDR) requires a post-market surveillance plan and periodic safety update reports, which provide information about good maintenance practices in high-risk contexts (pointing towards continuous monitoring activities).

Release versions should be archived with associated data, documentation, and simulation logs for test cases (NASA-HDBK-7009[11]). Version control is critical for accurate interpretation, repeatability, reproducibility, and debugging of the simulation predictions (Erdemir et al., 2020), and thus for the model's credibility. Ideally, automated tests (see Sect. 3.2.4) are run for each version in continuous integration setups and a standard workflow for releases is defined in continuous deployment setups (NASA-STD-7009A[12]).

As a good practice, irrespective of the model risk, new release versions require an additional verification and validation iteration by the developer to guarantee that the released version of the model sustains its credibility for the CoU. In high-risk contexts, continuously monitoring the model's capability to deliver credible results, recording incidents for analysis, taking corrective, adaptive, perfective, and preventive actions, and confirming restored capability (ISO/IEC/IEEE 12207:2017) are likely indicated.

End-of-life decisions will eventually be taken. It is important to inform the users in good time about the supported time frame for the model and to clearly communicate which support and training measures are available for users during which phases of the life cycle. Released versions must be archived and preserved in a format that permits execution beyond the supported lifetime. One solution can be software containers (e.g., Docker) that include all dependencies and only rely on an abstract execution layer that will be supported for an extended period. Also, model disposal measures prevent out-of-date model versions from returning to the supply chain ISO/IEC/IEEE 12207:2017 unless explicitly required.

[11] https://standards.nasa.gov/standard/nasa/nasa-hdbk-7009.

[12] https://standards.nasa.gov/standard/nasa/nasa-std-7009.

Table 3.6 Good practices for the maintenance phase of model development on the low-risk and high-risk ends of the risk spectrum

	Low risk	High risk
Maintenance	Execution of the maintenance strategy in the model development plan For each release, report the model's capability to deliver credible results archived together with associated data, documentation, and simulation logs for test cases	Additionally: –Continuous recording of incidents, taking corrective, adaptive, perfective, and preventive actions and confirming restored capability according to ISO/IEC/IEEE 12207:2017

3.3 Conclusions

In this Chapter, we outlined an approach to guide model development best practices based on a given CoU. Notice that this Chapter did not address the best practice for the end user of the model directly. Numerous industry standards guide on how to plan, implement, test, and maintain software, as part of medical devices and thus in critical, regulated environments. As mathematical models for healthcare often take the form of software, the application of an adapted industry standard from software development, for example, ISO/IEC/IEEE 12207:2007 or ICEI/IEC 62304:2006, seems possible. However, full compliance with industry standards is not always required or advisable. We, therefore, suggest using model risk (as defined in Chap. 4) to guide the stringency and level of adherence to industry standards. As a best practice, all models should comply with minimum requirements to anticipate that maximising compliance helps with model/software/platform qualification/certification in regulatory processes.

Life cycle planning reported by a model development plan is suggested as a critical step before implementation. Templates (e.g., SDP for medical device software in regulated environments (The Software Sustainability Institute, 2018)) are available. They can help to set up high-risk models compliant with current and future regulatory requirements. Requirements must be derived from a detailed analysis of the CoU, the mission and user need and then documented and traced throughout the development.

Of particular importance are the documentation of the model formulation and architecture design decisions, the design itself, and interfaces derived from the requirements as part of a model design document, including a description of intended use cases. Good software development practices should be followed during model implementation and integration, such as version control and the provision of tested code and end-user documentation.

During development and maintenance (as defined in the development plan), integration and systems testing should be performed and reported systematically and automatically. More involved testing paradigms (e.g., unit tests) and continuous monitoring must be

envisaged for high-risk environments. Also, testing confirming model credibility for the CoU must be repeated and reported with every model release in such environments.

This set of good model development practices provides a general, yet hopefully tangible, framework that applies to a wide range of in silico models and CoUs spanning different risk levels.

3.4 Essential Good Simulation Practice Recommendations

- Establish your model's CoU(s), related risks and requirements in a *model development plan* before defining and implementing the model (Table 3.2).
- Identify relevant industry standards for your model (Sect. 3.2).
- When designing the model for your CoU(s), consider relevant domain-specific standards, parameter identifiability and options for software verification and validation. Document the decision-making process for the conceptual model and the resulting limitations in the *model design document* (Table 3.3).
- Implement the model software based on established good practices for software engineering and development (Table 3.4) and follow a test-driven development paradigm (Table 3.5).
- Consider the entire model life cycle in the model management plan and secure adequate resources for maintenance (Table 3.6).

References

Anzt, H., Bach, F., Druskat, S., Löffler, F., Loewe, A., Renard, B. Y., Seemann, G., Struck, A., Achhammer, E., Aggarwal, P., Appel, F., Bader, M., Brusch, L., Busse, C., Chourdakis, G., Dabrowski, P. W., Ebert, P., Flemisch, B., Friedl, S., Fritzsch, B., Funk, M. D., Gast, V., Goth, F., Grad, J.-N., Hegewald, J., Hermann, S., Hohmann, F., Janosch, S., Kutra, D., Linxweiler, J., Muth, T., Peters-Kottig, W., Rack, F., Raters, F. H. C., Rave, S., Reina, G., Reißig, M., Ropinski, T., Schaarschmidt, J., Seibold, H., Thiele, J. P., Uekermann, B., Unger, S., & Weeber, R. (2021). An environment for sustainable research software in Germany and beyond: Current state, open challenges, and call for action. https://doi.org/10.12688/f1000research.23224.2

Bäker, M. (2018). How to get meaningful and correct results from your finite element model. arXiv: 1811.05753.

Byon, W., Smith, M. K., Chan, P., Tortorici, M. A., Riley, S., Dai, H., Dong, J., Ruiz-Garcia, A., Sweeney, K., & Cronenberger, C. (2013). Establishing best practices and guidance in population modeling: An experience with an internal population pharmacokinetic analysis guidance. *CPT: Pharmacometrics & Systems Pharmacology, 2*, e51. https://doi.org/10.1038/psp.2013.26

Chue Hong, N. P., Katz, D. S., Barker, M., Lamprecht, A.-L., Martinez, C., Psomopoulos, F. E., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., & Honeyman, T. (2021). FAIR principles for research software (FAIR4RS principles). Research Data Alliance. https://doi.org/10.15497/RDA00068

Cucurull-Sanchez, L., Chappell, M. J., Chelliah, V., Amy Cheung, S. Y., Derks, G., Penney, M., Phipps, A., Malik-Sheriff, R. S., Timmis, J., Tindall, M. J., van der Graaf, P. H., Vicini, P., & Yates, J. W. T. (2019). Best practices to maximize the use and reuse of quantitative and systems pharmacology models: Recommendations from the United Kingdom quantitative and systems pharmacology network. *CPT: Pharmacometrics & Systems Pharmacology, 8*, 259–272. https://doi.org/10.1002/psp4.12381

Erdemir, A., Besier, T. F., Halloran, J. P., Imhauser, C. W., Laz, P. J., Morrison, T. M., & Shelburne, K. B. (2019). Deciphering the "Art" in modeling and simulation of the knee joint: Overall strategy. *Journal of Biomechanical Engineering, 141*, 0710021–07100210. https://doi.org/10.1115/1.4043346

Erdemir, A., Mulugeta, L., Ku, J. P., Drach, A., Horner, M., Morrison, T. M., Peng, G. C. Y., Vadigepalli, R., Lytton, W. W., & Myers, J. G. (2020). Credible practice of modeling and simulation in healthcare: Ten rules from a multidisciplinary perspective. *Journal of Translational Medicine, 18*, 369. https://doi.org/10.1186/s12967-020-02540-4

Galappaththige, S., Gray, R. A., Costa, C. M., Niederer, S., & Pathmanathan, P. (2022). Credibility assessment of patient-specific computational modeling using patient-specific cardiac modeling as an exemplar. *PLOS Computational Biology, 18*, e1010541. https://doi.org/10.1371/journal.pcbi.1010541

Jean, D., Naik, K., Milligan, L., Hall, S., Mei Huang, S., Isoherranen, N., Kuemmel, C., Seo, P., Tegenge, M. A., Wang, Y., Yang, Y., Zhang, X., Zhao, L., Zhao, P., Benjamin, J., Bergman, K., Grillo, J., Madabushi, R., Wu, F., Zhu, H., & Zineh, I. (2021). Development of best practices in physiologically based pharmacokinetic modeling to support clinical pharmacology regulatory decision-making—A workshop summary. *CPT: Pharmacometrics & Systems Pharmacology, 10*, 1271–1275. https://doi.org/10.1002/psp4.12706

Ke, A., Barter, Z., Rowland-Yeo, K., & Almond, L. (2016). Towards a best practice approach in PBPK modeling: Case example of developing a Unified Efavirenz model accounting for induction of CYPs 3A4 and 2B6. *CPT: Pharmacometrics & Systems Pharmacology, 5*, 367–376. https://doi.org/10.1002/psp4.12088

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE, 68*, 1060–1076. https://doi.org/10.1109/PROC.1980.11805

Overgaard, R., Ingwersen, S., & Tornøe, C. (2015). Establishing good practices for exposure-response analysis of clinical endpoints in drug development. *CPT: Pharmacometrics & Systems Pharmacology, 4*, 565–575. https://doi.org/10.1002/psp4.12015

Parvinian, B., Pathmanathan, P., Daluwatte, C., Yaghouby, F., Gray, R. A., Weininger, S., Morrison, T. M., & Scully, C. G. (2019). Credibility evidence for computational patient models used in the development of physiological closed-loop controlled devices for critical care medicine. *Frontiers in Physiology, 10*, 220. https://doi.org/10.3389/fphys.2019.00220

Pathmanathan, P., & Gray, R. A. (2014). Verification of computational models of cardiac electrophysiology. *International Journal Numerical Method Biomedical Engineering, 30*, 525–544. https://doi.org/10.1002/cnm.2615

Pathmanathan, P., Gray, R. A., Romero, V. J., & Morrison, T. M. (2017). Applicability analysis of validation evidence for biomedical computational models. *Journal of Verification, Validation and Uncertainty Quantification, 2*. https://doi.org/10.1115/1.4037671

Rust, P., Flood, D., & Mccaffery, F. (2016). Creation of an IEC 62304 compliant software development plan. *Journal of Software: Evolution and Process, 28*. https://doi.org/10.1002/smr.1826

The Software Sustainability Institute. (2018). Checklist for a software management plan. Zenodo. https://doi.org/10.5281/zenodo.1460504

Zhao, P., Rowland, M., & Huang, S.-M. (2012). Best practice in the use of physiologically based pharmacokinetic modeling and simulation to address clinical pharmacology regulatory questions. *Clinical Pharmacology & Therapeutics, 92*, 17–20. https://doi.org/10.1038/clpt.2012.68