

Chapter 1

Introduction

Secure system development is not a trivial task. It comprises a number of activities, which need to be combined, analysed, and executed to produce a secure software system. In this section, after defining secure system engineering we give an overview of few existing security and security risk management standards. The chapter is followed by a survey of security modelling approaches, domain-specific languages for security modelling and principles of model-driven security. An important concern of the software system is the ability to capture multiple system perspectives. In this chapter we survey various modelling perspectives and argue for the importance of capturing these perspectives and complementing the system models using model transformation principles. This chapter also presents a running example and the structure of this book.

1.1 System and Security Engineering

In information system theory, a “*system* is a set of correlated phenomena, which itself is a phenomenon” [109]. However, in practical settings we could understand system [15] as (i) a product or component (e.g., personal computer, smartcard, a piece of software, etc.), (ii) infrastructure needed to combine the products or components (e.g., operating system, network, etc.), (iii) applications that are used to support everyday activities (e.g., browser, financial system, word processor, etc.), (iv), information technology staff who support the above-mentioned components, (v) internal users and management, who uses the technology to produce business products or services, and (vi) customers and other external users, who buy products and use services. In other words, *system* is everything, since all these elements (i.e., phenomena) correlate to each other. In addition, the system will not be a stand-alone element, but will interact with other systems; thus it will operate in the environment, making other systems on its own.

“*Security engineering* is concerned with lowering the risk of intentional unauthorised harm to valuable assets to a level that is acceptable to the system’s stakeholders by preventing and reacting to malicious harm, misuse, threats and security risks” [70]. Hence, security engineering is dealing with *intentional unauthorised risks* and it is different from *safety* engineering, where *unintentional* risk is considered. During security engineering process, analysts must understand what stakeholders’ values (in terms of assets) are and how these values must be protected. It is not possible to reach “100 percent security” because there exist feasibility and acceptability boundaries defined during the engineering process. Finally, the definition highlights that security risk could be determined in different forms, including harm, misuse, and threat. Thus, the task of the security analyst is to recognise these forms and uncover the complete picture of the security risk.

Security engineering concentrates on tools, processes and methodologies that support analysing, designing and implementing new systems or adjusting an existing system according to the needs of its environment. The analysis of security should be performed through the whole software development process, starting from early requirements and going through later requirements, design, and implementation stages. An early consideration of security allows analysts and modellers to discover threats, understand their consequences and develop countermeasures to mitigate these threats. It also proposes security design alternatives, and could result in the decision of project cancellation if the security risks are too high.

1.2 Security and Security Risk Management Standards

There exist a number of standards to manage security or security risks. In [1] a short overview of these standards is presented. In this section we will recall it to explain the important security and security risk management principles.

Security standards define guidelines for information security management. For instance, the ISO/IEC 13335-1:2004 standard [93] defines security concepts and models fundamental for a basic understanding of information and communication technology security. It also addresses the general management issues that are essential to the successful planning, implementation and operation of information and communication technology security. The Common Criteria for information technology security evaluation [36] standard specifies security requirements and the desired security goals. The goal of evaluators is to determine whether the chosen security requirements help to achieve the defined security measures and to check whether they are implemented correctly.

The *security risk management standards* particularly focus on security risk management activities. Following [1, 133] we include a short overview of the ISO/IEC 2700x series [94, 95], a NIST special publication [154, 155], and the BSI standards 100 series for information technology [32–35].

The ISO/IEC 27001:2013 standard [95] defines the requirements to establish, implement, maintain, continuously improve and manage an information security

management system. It presents requirements for managing an organisation's sensitive information by applying risk management, risk assessment and risk treatment means. The major parts of the standard include guidance on understanding context of organisation, leadership, planning, support, operation performance evaluation and improvement activities. It also provides a checklist of objectives and controls (although an organisation could also choose other controls according to its preferences). The objectives and controls could be combined to develop an organisation's treatment plan to respond to security risks. The ISO/IEC 27001 is supported by ISO/IEC 27005 [94], which introduces the security risk management process. The process consists of context establishment, risk assessment, risk treatment, risk acceptance, risk communication, consultation, risk monitoring and review. The process uses an iterative approach for risk assessment and/or risk treatment activities.

The National Institute of Standards and Technology (NIST) has published a set of standards addressing security and risk in information systems. For example, according to the NIST SP 800-39 standard [154], the risk management approach consists of four components: (1) frame risk; (2) assess risk; (3) respond to risk; and (4) monitor risk. The risk assessment components are addressed in NIST SP 800-30 [155]. This standard introduces a process consisting of risk assessment preparation, risk assessment conduction, risk assessment result communication, and risk assessment maintenance. In other words, the process guides the communication between the risk assessment and other organisational risk management processes.

An IT-Grundschutz is series of German standards for security management methods. It includes four standards for information security management systems [32], methodology [33], risk analysis [34], and business continuity management [35]. The standard on information security management systems [32] specifies the requirements of such systems. The methodology standard [33] defines successive steps to guide a management system for information security. It describes how to introduce and maintain an information security policy, how to select information security safeguards to implement the information security policy, and how to maintain and improve information security during its operation. The standard on risk analysis [34] highlights the risk analysis methodology in addition to the methodology defined in the second standard. A supplementary security analysis should be done for the particular assets (or target objects) determined in the methodology. The fourth standard on business continuity management [35] is based on the previous standards. It presents a method to detect the risks that can endanger the survival (i.e., economic existence) of an organisation, and to develop safeguards against such risks or even after incurring the risks. Overall, the BSI standards present a process for achieving and maintaining the required level of security, and the way to determine and achieve it.

The standards give an explanation of components to define the security and security risk management domain model. In Chap. 2 the domain model for security risk management is discussed. This model combines the principles of the above-mentioned standards to the systematic perspective and guidance for the security requirements definition and reasoning.

1.3 Security Development Approaches

There exist few literature surveys which present extensive overviews and comparisons of security requirements techniques, frameworks, processes and methodologies. For instance, in [63], Fabian et al. present a conceptual framework to compare and evaluate security requirements engineering approaches. In Chap. 4 a number of the security development approaches are summarised following the technique, framework, process and method classification suggested in [143].

In this book we will look at a few secure system modelling approaches especially ones related to system security risk management. For instance, we consider how business process model and notation can support security risk management. Next we illustrate extensions of Secure Tropos to security risk-aware analysis. To capture functional and behavioural properties of secure software systems, we apply misuse cases and mal-activities. UMLsec [100] and SecureUML [21, 119] will be considered to define access control authorisation constraints. Finally we illustrate the security requirements elicitation from business processes (SREBP), an approach to derive security requirements from the BPMN models [1, 4, 178].

1.4 Domain-Specific Languages for Security Modelling

Domain-specific modelling languages focus on particular a domain and help us find and visualise the targeted solution for problems of this domain. As in other domains, there exist a number of domain-specific languages for security modelling and engineering. For instance, approaches to model security using domain-specific languages are reported in [30]. Elsewhere in [122], the CORAS modelling language is dedicated to security and safety risk analysis. It suggests visual constructs to decompose and analyse threats, to reason and to introduce security countermeasures. In [195], the cyber security modelling language (CySeMoL) is introduced to express enterprise-level system architectures and to capture security attacks and reason about the countermeasures.

In this work we will consider extensions of traditional modelling approaches—BPMN [54, 159, 189], Tropos [29], misuse cases [158, 192], and mal-activity diagrams [158, 191]—and illustrate how they could be extended to the domain-specific modelling languages. This helps us manage security risks. We will also consider the domain of role-based access control and modelling languages—SecureUML [21, 119] and UMLsec [100]—dedicated to capturing principles of the role-based access control.

1.5 Model-Driven Security

Model-driven development is a development paradigm which uses models through the development cycle. Models could be used for the different purposes (see Fig. 1.1), including code visualisation, base modelling, roundtrip engineering, and model-centric development [197]. Models bring the abstraction of the real world to simplified viewpoints, and focus on certain concerns or phenomena. Models help transform code-based decisions to model-based decisions, without influencing the level of abstraction captured in the considered software system. Models also help separate classes of phenomena needed to implement the software system.

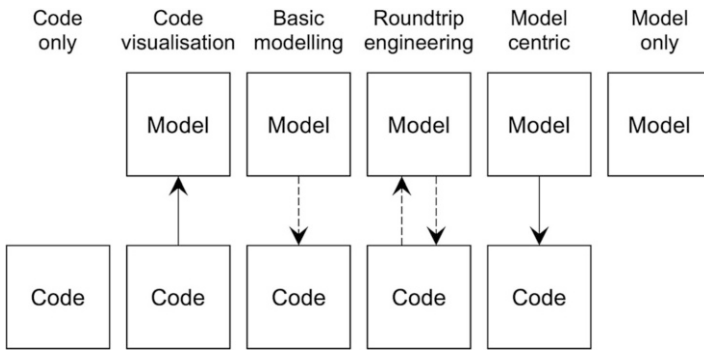


Fig. 1.1 Links between code and model, adapted from [197]

Models can be used for various reasons, but one of the principles of model-driven development concerns the means for model transformation. In [214] a taxonomy of transformation approaches is presented (see Fig. 1.2). Rule-based transformations are performed following predefined rules. Ontology-based transformations share vocabulary, where a common concepts are defined; hence the ontology acts as the intermediate level to transform models from one language representation to another based on the construct semantics. Identify-based transformations translate the source model to the target model without changing information: “the two models describe the same concepts but with different representation” [214]. The pattern-based transformation translates the source patterns (set of source elements) to target patterns (a set of target elements).

Model-driven security is an application of the model-driven technologies to develop secure software systems. In principle it defines the transformation-oriented approach to generate security constraints from the security requirements models. Principles of model-driven security could be used for various purposes. In this book we will discuss the use of models to elicit security requirements through security risk management. Next we will use models to represent various security requirements perspectives, including functional, behavioural, actor and role, goal and rule, and topological perspectives. We will apply rule-based and ontology-based trans-

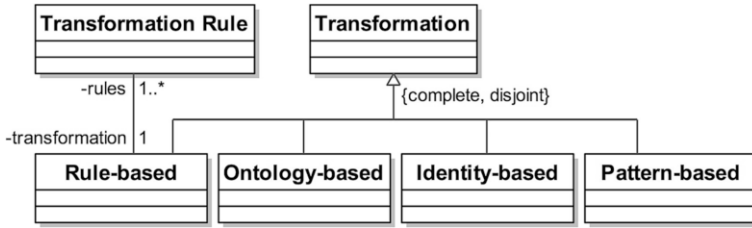


Fig. 1.2 Transformation taxonomy, adapted from [214]

formation methods to capture these different security perspectives of both security requirements and secure system design. We will also illustrate pattern-based transformation to capture security constraints from models of the secure software system.

1.6 Modelling Perspectives of Secure Software Systems

Software development typically involves different stakeholders with different goals, needs, requirements and system expectations [157]. Addressing all the different viewpoints and agreeing about them requires using various modelling techniques, estimating their dependencies, and managing them all in a consistent and coherent manner. In [109] Krogstie isolates among eight modelling perspectives: behavioural, functional, structural, goal and rule, object, communication, actor and role, and topological perspectives. The *behavioural perspective* deals with the phenomenon of states and transitions between them. The *functional perspective* defines a transformation as an activity, which transforms one set of phenomena to a different set of phenomena. The *structural perspective* concentrates on the static structure of the software system; it deals with entities, their relationships, their attributes and their values. The *goal and rule* perspective focusses on modelling goals (something that needs to be achieved) and rules (something that needs to be satisfied). The *object perspective* defines the main phenomena as objects, processes, and classes. The *communication perspective* is based on the language/action theory and on the premise of human collaboration through communication. The *actor and role perspective* originates from organisational modelling and deals with actors, agents, roles and their dependencies. Finally, the *topological perspective* deals with ordering between different perspectives.

Examples of Modelling Perspectives Although most of the modelling approaches cover several modelling perspectives at the same time, typically they acquire one primary perspective. For instance, business process model and notation (BPMN) [54, 159, 189] and UML activity diagrams [158] belong to the *functional perspective*. They both include the major construct groups such as flow objects, swimlanes, artefacts and connecting objects. However, BPMN is rather meant to model an organisation’s workflows and business processes; in other words, BPMN is used

to visualise *business* entities, their collaborations, and data flow and describe how the software system can be used to achieve business goals. The activity diagrams are dedicated to the dynamic aspects of the considered *software system*. In other words, activity diagrams are used to express the functional perspective of the software system and/or software system components.

UML use case diagrams are typically used to present the *behavioural perspective*. A use case represents “a declaration of an offered behaviour” [158], which is characterised by activities, interactions and states as well as pre-conditions and post-conditions.

Another modelling approach considered in this book is Tropos [29]. It originates from the *i** modelling language [213] used to capture social relationships between organisation’s actors. Hence actor could be classified as agent, position or role. Agent could occupy position, position could cover role and agent could play the role. Naturally, the *i** and Tropos modelling approaches capture the *actor and role* modelling perspective, where the dependency relationships are defined between different agents (including the software system) in the organisation. Tropos also covers the *goal and rule* perspective, as one of the purposes of modelling using Tropos is to elicit and satisfy an actor’s goal through the eligible performance of the plans and use of the available resources.

In the second part of this book we will consider how different modelling approaches (i.e., BPMN, Secure Tropos, UML use cases, and UML activity diagrams) can help in managing security risks using the language alignment to the domain model of the information systems security risk management [53, 133]. This alignment will illustrate the *topological perspective*, where different modelling constructs will be identified to address certain concepts of the security risk management domain.

Capturing Various Perspectives There were a few attempts to support a smooth transition between different the security modelling perspectives. For instance, [80] presents MEMO, a multi-perspective enterprise modelling. Its core components support multi-perspective development of secure systems. It concentrates on the technological, organisational and economic perspectives, takes into account the relevant contents, and provides integration of the perspectives. Hence the security risk management consists of four activities: (i) enterprise characterisation (i.e., asset identification, definition of goals and business structure, and others), (ii) security risk identification (i.e., identification of security threats and vulnerabilities), (iii) security risk evaluation (i.e., evaluation of threat likelihood and impact and determination of security risks), and (iv) security risk mitigation (i.e., risk prioritisation, control recommendation and implementation). Although MEMO suggests modelling notations to capture various perspectives and define their consistent representation and traceability, it still remains a domain-specific modelling approach that needs to be learnt by the stakeholders.

Another approach to linking secure system modelling perspectives is through the transformation of different models, as illustrated in Fig. 1.3. Such an approach has few advantages. Firstly, it helps capture the security concerns using different modelling languages, which still could (at least one or few representations) be

familiar to the various stakeholders. Thus, the stakeholders would not need to learn new notations, but be familiar with the standard ones.

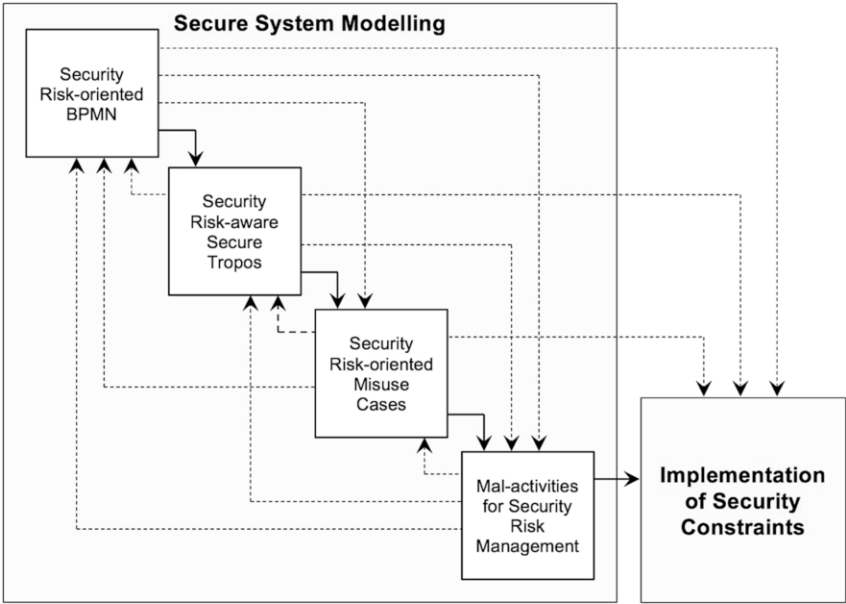


Fig. 1.3 Multiple modelling perspectives: approaches described in this book (*solid arrows* describe transformations discussed in this book, *dashed arrows* - other possible transformations)

Secondly, it helps ensure a systematic and consistent security engineering process through different development stages. This also helps maintain traceability between the security requirements, their implementation in the security controls and their transformation to the security constraints in the developed software application.

Thirdly, the transformations help to enhance the model of the considered system. As these modelling approaches, following the topological perspectives, would be aligned to the common ontology (in our case this is the domain model for security risk management [53, 133]), the model translations would be performed using ontology-based transformations.

As illustrated in the third part of the book, typically, the transformed model will not capture the same information that is defined in the original model. The modeller would complement the new model with additional details, bringing additional details and capturing the more complete description of the analysed system. Additionally, the transformed model would stimulate further system development with additional security concerns and help us see new emerging security risks and their countermeasures.

Finally, the security transformation could result in a transformed security code, which would help us capture security constraints of the software system. In the book,

the principles of pattern-based transformation from security models (expressing the RBAC policy) to the security constraints will be illustrated.

1.7 Running Example

General Description In this book a *Football Federation case* is used as an example to show various principles of secure system modelling. This example was partially analysed in [107]; however, in this book a hypothetical case is developed to illustrate different principles of secure system modelling and design. Figure 1.4 provides some informal description of the system used in the *Football Federation case*. Like all organisations, Football Federation has a number of employees. For example, Football Federation Employee is carrying the typical business workflows in the organisation. To support his business activities he is using various systems. One of them is Electronic Registration Information System (ERIS), typically consisting of User Interface, Information Processing System, and Database, as illustrated in Fig. 1.4.

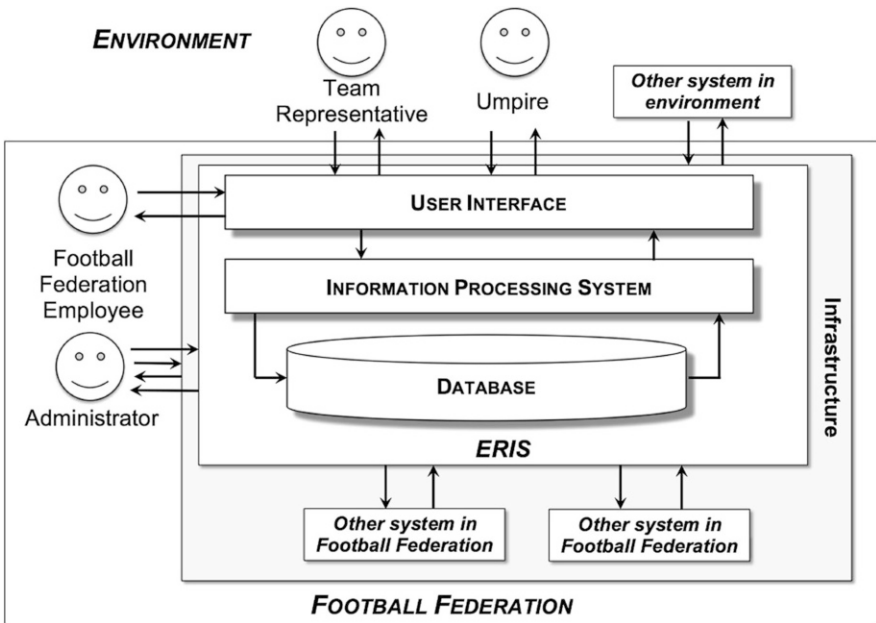


Fig. 1.4 Informal description of the Football Federation case

ERIS is not a standalone application, but is connected to **Other systems** used within the Football Federation organisation, and also **Other systems** existing in the environment. To maintain the Football Federation systems, a certain **Infrastructure** (including operating systems, networks, connection channels, etc.) should be estab-

lished and constantly maintained. This is responsibility of an **Administrator**, who is an employee of the Football Federation.

Additionally there might exist *external users*, such as **Team Representative** and **Umpire**, who potentially might access the ERIS for one or another service or needed information. The Football Federation organisation is not isolated, and, as mentioned above, is connected to **Other systems** existing in the environment, which could request some services from ERIS or provide services by themselves.

Conceptual Model An extract of the conceptual model of the ERIS system is presented in Fig. 1.5 as a UML class diagram. It includes five concepts: **Game**, **Team**, **Player**, **Timetable** and **Umpire**.

Game represents games organised and administered by the Football Federation. Games are described by their information (i.e., attribute **gameInfo**), assigned umpired (i.e., attribute **umpire**), game report (i.e., attribute **gameReport**), and confirmation of the game report (i.e., attribute **confirmation**).

Team describes the football teams which are members of the Football Federation. Each team is characterised by its information (i.e., attribute **teamInfo**, which might include name, city, sponsor, etc.). Team has a team representative (i.e., attribute **teamRep**). Team makes a decision about participating in the tournaments (i.e., attribute **participationDecision**). Finally, if the participation decision is positive, the information about the team's region and league is added (i.e., attribute **regionAndLeague**).

Players are the members of the team. In this model, each player is described by his information (i.e., attribute **playerInfo**, which might include name, surname, position in the team, etc.) and his performance in a certain game (i.e., attribute **playerPerformance**).

In **Timetable**, a schedule of all games administered by the Football Federation is assembled. Timetable is described using timetable information (i.e., attribute **timeTableInfo**, list of including leagues, playing teams, place, etc.) and schedule of games (i.e., attribute **schedule**). In addition, each timetable should be confirmed by the league administrators (i.e., attribute **timeTableConfirmation**).

Finally, each game is assigned a certain number (typically 3, 4) of **Umpires**. Each umpire is described by his umpire info (i.e., attribute **umpireInfo** including umpire's name, surname, expertise, etc.). Umpire also receives special access to the ERIS (i.e., attribute **umpireAccess**), so that he can provide information about his games (described using the **assignedGames** attribute).

Business Processes The BPMN *value chain diagram* describes how an organisation's business functions are related to each other in order achieve the business goals [54, 159, 189]. In Fig. 1.6 an extract of the ERIS value chain is provided. The selected subprocesses are

- **Register team**—registering teams to the ERIS system;
- **Register player**—registering players and assigning them to the teams;
- **Register umpire**—registering umpires to the ERIS system;
- **Create leagues and divisions**—creating new leagues and preparing divisions;
- **Create timetable**—creating and preparing a timetable;

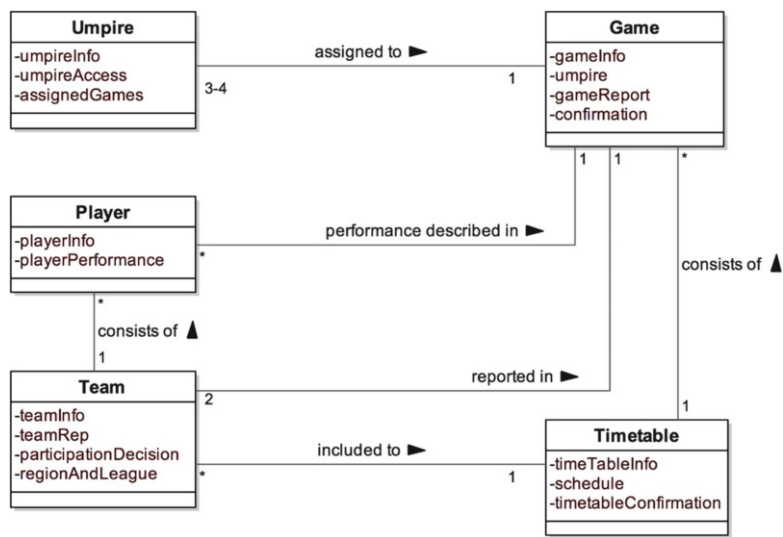


Fig. 1.5 Conceptual model

- Register game—registering games and their results.

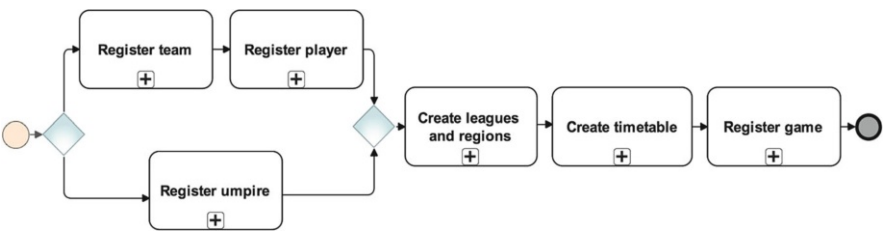


Fig. 1.6 Business process value chain

The latter subprocess—Register game report—is expanded in Fig. 1.7. Here the Football Federation Employee decides to enter a new game (see task Enter new game) and thus submits the game information (see task Submit game info); once the initial game information is received, the entry is created in the game storage (see task Create game). The Football Federation Employee also assigns the umpire (see task Assign umpire). After the game the Umpire submits a game report (see task Submit game report). After the game report is updated (see task Update game report) it needs to be confirmed by the Football Federation Employee (see task Confirm game report).

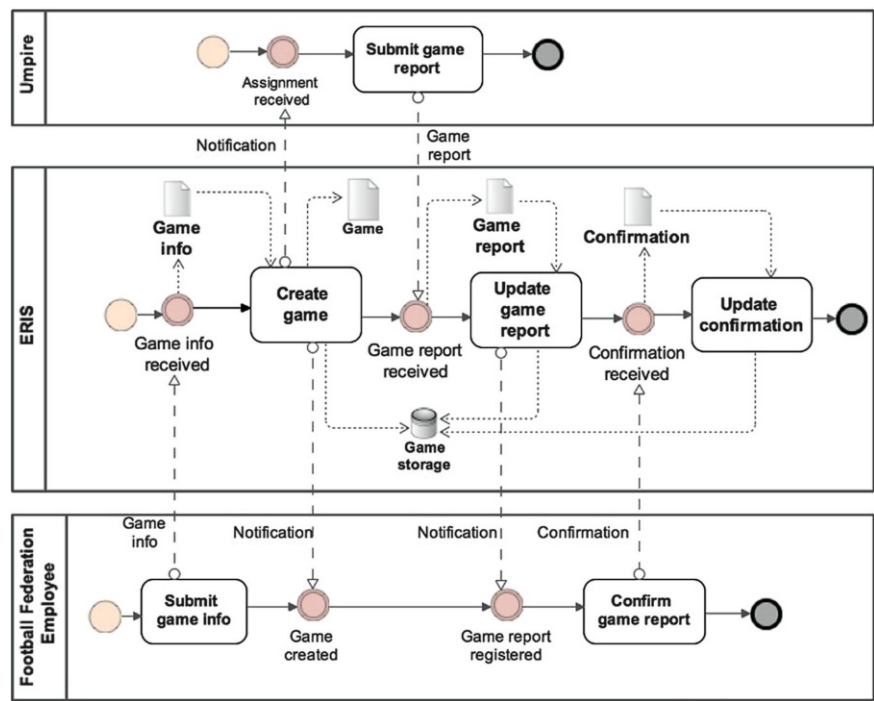


Fig. 1.7 Process to register game report

1.8 Exercises

Exercise 1.1 Describe what “system” is in the *Football Federation case*. What or who are the products/components of the system, its infrastructure applications, information technology staff, internal users and management, customers and other external users? To support your answer, fill in Table 1.1.

Table 1.1 Template to support answer of Exercise 1.1

System components	Football Federation case
Products / Components	
Infrastructure	
Applications	
Information technology staff	
Internal users and management	
Customers and external users	
Environment	

Exercise 1.2 Compare the components and processes of different security risk management standardisation initiatives:

- ISO/IEC 2700x vs. NIST;
- NIST vs. BSI;
- BSI vs. ISO/IEC 2700x.

Exercise 1.3 Discuss what techniques, frameworks, processes and methods for secure software system development are familiar to you from your previous practices and courses. What were your bad and good experiences?

Exercise 1.4 What are modelling perspectives? How do different modelling perspectives influence development of secure software systems?

Exercise 1.5 Discuss advantages and disadvantages of the transformation-driven approaches to developing secure software systems?

Exercise 1.6 How could different Football Federation subprocesses (see Fig. 1.6) be expanded? Model Register team, Register player, and Create leagues and regions subprocesses using business model and notation, BPMN (or any other modelling language of your preference).

Fundamentals of Secure System Modelling

Matulevičius, R.

2017, XX, 218 p. 112 illus., 62 illus. in color., Hardcover

ISBN: 978-3-319-61716-9