

# Security-Driven Information Flow Modelling for Component Integration in Complex Environments

Veronika Kupfersberger  
University of Vienna  
Vienna, Austria  
veronika.kupfersberger@univie.ac.at

Thomas Schaberreiter  
University of Vienna  
Vienna, Austria  
thomas.schaberreiter@univie.ac.at

Gerald Quirchmayr  
University of Vienna  
Vienna, Austria  
gerald.quirchmayr@univie.ac.at

## ABSTRACT

Conceptualising and developing a new software solution is always a daunting task, even more so when existing technologies of international partners are to be integrated into a unique and holistic product, as is the case in many international research and innovation projects. The individual requirements not only of each tool, but of the resulting solution as a whole, must be considered as well as the problem domain. The approach presented in this paper uniquely combines existing structuring and modelling techniques, resulting in an information flow model and interface definition specifications appropriate for international projects. It is based on an approach developed for an EU cybersecurity project and for its specific requirements, but due to its flexibility seen as appropriate for other domains as well. Complex systems consisting of different existing software solutions are represented in a conceptual model of their internal processes and the connecting information flows, thereby facilitating further software development and adaptations. Additionally, the exact identification and accounting of all information flows are essential requirements for modelling according to security and privacy by design principles, as for example prescribed by privacy and impact assessment guides and required by the General Data Protection Regulation (GDPR).

## KEYWORDS

Security by Design, Complex Systems, Security Model, Information Flow Model, Framework Design, Software Component Integration

## 1 MOTIVATION AND BACKGROUND

Good software architecture design is the base of any successful information technology application [2]. When designing a new software solution, the preconditions vary based on the objectives as well as the environment in which it is to be developed and deployed. Possibilities include, but are not limited to projects designed from scratch, adapted legacy systems or systems combining existing solutions. A new combination of existing solutions is the main focus of this paper.

The approach presented in this paper is based on the following setting of the EU project during which it was developed:

- An international consortium of academic and industry partners
- New European legislations to consider (GDPR, NIS)
- The general socio-technological nature of the cybersecurity problem domain
- The complexity of component integration in a multi-stakeholder context

This environment has strongly influenced the choices made regarding the selection of the analysis and modelling methods, all of which will be described in more detail in the following chapter.

According to Saitta [18], complex systems are defined as systems including multiple components which do not influence the system itself, but their interactions do, and are typically a concept used in physics and art. They can be easily traversed into a construct in computer science as well and are further detailed in Section 2. Due to the complexity and variety of software architecture development processes, the procedure of abstraction, particularly when dealing with complex systems, is crucial to produce a cohesive model [10]. A new factor influencing the complexity of such systems are the regulations on Data Protection (GDPR) [12] and Network Information Security (NIS) [13] in the European Union, which strongly influence the requirements to information flows in complex systems.

Two concepts that are enforced by this new legislation are security and privacy by design and by default, which shifts and expands many of the requirements for software security from the operational phase to the design phase of products. For the first time secure software design becomes a legal requirement, which in turn leads to a re-thinking of currently practiced design methodologies, where security and privacy are often only an afterthought. Due to the complexity imposed by security and privacy on software design, a purely technological approach to design seems increasingly inadequate. That is why a socio-technological approach is needed that is able to capture the complex social and technological interactions that contribute to the security of software. In security by design, the main incentive is to identify the weak or vulnerable points a malicious actor can use to compromise the system and identify a more robust design. In privacy by design, the main incentive needs to be to determine and specify the information flows of personal and sensitive data through the system, to identify weak points that would allow unintended use of personal/sensitive data, and account for adequate data protection mechanisms in the design. We argue that this identification can only be achieved if all the stakeholders of a complex system (e.g. customers/users, designers, implementers, security/privacy experts) work together in order to elicit and address the security/privacy related aspects of the system and thus achieve security and privacy by design.

The approach to modelling complex system and their information flows presented in this paper was advanced during the conceptualization of the CS-AWARE information flow model and the definition of the respective interfaces as described in Section 4. CS-AWARE is an EU-funded Horizon 2020 project for innovative actions aimed at developing a cybersecurity awareness solution for local public administrations (LPA) in the European Union and

provides the case study examples for this paper. The solution combines existing tools developed by various partners, which required a well-structured and coherent model of information flows as an implementation and software development basis to facilitate the subsequent work packages. The result of the project is a monitoring tool, which analyses various input data and raises awareness in regards to incidents and potential threats among the IT administrators. While the approach was designed specifically for the cybersecurity domain, the socio-technical analysis and the modelling of the information flow can be easily used for software component integration.

When designing a meaningful software solution based on existing tools, the first step is to identify the capabilities and responsibilities of each component and how they are linked before integrating them to receive a new software solution. This is particularly important when developing monitoring software, because the monitored entities do not only perform different tasks but also provide various types of data. The final product can be divided into two sub-entities: the monitoring tools and the technologies to be monitored, the latter of which need to be integrated into the final model as well. The relevance of specifying information flows to ensure security of the system has been discussed and incorporated in other modelling approaches [17]. This is especially challenging when combining technologies by international providers from various sectors. The optimal information flow between the respective software entities must be thoroughly analysed and defined before continuing with the technical adaptations and the software development. Security and privacy by design are ensured by carrying out detailed specifications of information flows only after a thorough analysis of the domain.

The selected analysis and design methods are described in the Section 2 on related work. Section 3.1 covers the conceptualization strategies relevant for constructing an information flow model useful to the software architecture. In the subsequent Section 3.2, a modelling technique for information flow models of complex systems will be described in detail, after which a brief introduction to the case study on which this approach was first applied is given in Section 4. The paper ends with a conclusion on the presented approach to modelling complex systems and the outlook on future works in Section 5.

## 2 RELATED WORK

When developing a new software it is essential to have a holistic idea of the problem it aims to solve and the respective domain. When dealing with a socio-technical domain, such as cybersecurity, where human actions and tasks are as relevant as any technical process, Soft Systems Methodology (SSM) [3] is an appropriate choice. Other structuring methods, such as cognitive mapping, casual loop diagrams [15] or a combination of stakeholder analysis and cognitive mapping as suggested by Ferretti [6], would have been an alternative. SSM was chosen for this particular approach due to the existing knowledge in the consortium and the explicit socio-technical nature of the methodology.

With SSM, the problem definition is supported by the various stakeholders who create rich pictures describing different aspects of the domain in a guided workshop. This allows a much deeper

insight into the domain and its issues, since it combines different views and experiences. The seven steps of the SSM are:

- Entering the problem situation,
- Expressing the problem situation,
- Formulating root definitions of the systems behaviour,
- Building conceptual models of the systems,
- Comparing models to real-life situations,
- Defining possible changes and
- Taking action to improve the problem situation [3].

The result of these workshops are evaluated by the analyst and potentially external experts and used as input for the conceptualisation of the model. This method helps to simplify complex integrations of different components into one new software. Complex systems in software engineering are systems where components function independently but are strongly influenced by results of others [8] [18]. According to Jiang et al. [8], these complex software systems typically involve encapsulated software components that interact with each other. When combining existing software solutions into one, this can often be the case. Therefore, these complex systems must be simplified and abstracted in such a way that a useful, coherent and holistic model can be generated. Since there is no definition of abstraction widely recognized that covers more than ‘... *the generic idea of distilling the essential*’ [18], also approaches in the disciplines vary. Abstraction in software engineering can occur in two ways, according to Sokolowski et. al [19] either by limiting the information covered by the model to only the components which are relevant and ignoring the remaining or by reproducing a minimized version of the real-world concept. This procedure of abstraction is critical and sometimes considered one of the most important capabilities of a software engineer [10].

As the communication between the components is such an integral part of a complex system, the model can be considered an information flow model of such a system. Information flows can cover multiple granularities of interconnections between components, but on a high-level can be classified in three categories [4]:

- *Direct information flow*  
Component A communicates with component B directly with no other components involved.
- *Indirect information flow*  
Component A and component B share information and data via component C. All communication shared between A and B go through C, which means A and B indirectly communicate with C as well.
- *General information flow*  
Either one of the two options exist between two components, so any type of information flow is possible.

The modelling of the information flow is based on the Data Flow Diagram (DFD) language defined by Chen [11] but adapted to suit the domains needs. The types of data flows have been altered, while types of activities were added to ensure that the diversity of the system can be modelled easily. This approach was chosen due to the strong focus and importance of the information flow

in the CS-AWARE solution as well as the need for individualised entities. While UML (Unified Modelling Language) activity and class diagrams are superior for modelling architectural components, information flow modelling has its own characteristics. That is why we chose to build an extension to the DFD language allowing us to represent certain aspects of the information flows in a more comfortable way, especially when modelling complex systems.

When dealing with complex systems, all types of the above information flows are likely to occur and must be identified before constructing the model of the system. Next to abstraction, the abstraction level on which to design such software models needs to be identified. In the field of software design and simulation modelling conceptual modelling can be defined as "... *the process of abstracting a model from a real or proposed system* [14]". It is essential to design conceptual models as broadly as possible, in an iterative and continuing approach [16]. Lacking alternate guidelines to conceptual modelling, Robinson [14] [15] defined four requirements of a conceptual model (validity, credibility, utility and feasibility) as well as a framework for applying conceptual modelling.

The problem domain influences the model repetitively, but itself receives input from the model. Defining general objectives of the project and the model are important tasks before collecting relevant input and output factors, since the latter are strongly influenced by the goals of the model. Input factors are of either qualitative or quantitative nature and the means with which the models' objectives can be achieved, while output factors are values with which to measure the efficiency and quality of the conceptual model. Finally, the scope of the model where any assumptions made are defined.

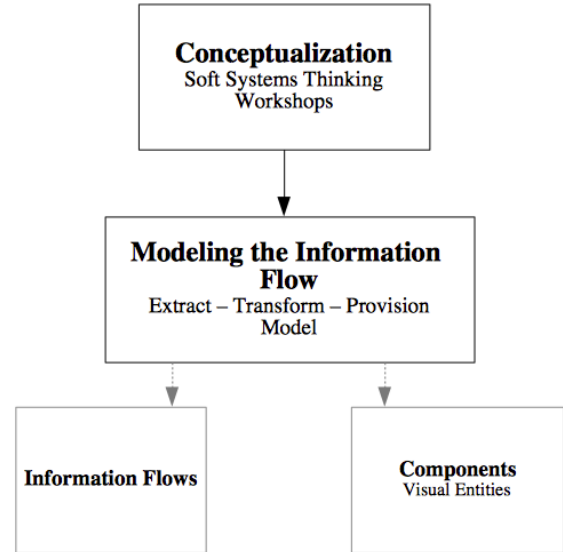
The modelling approach documented in this paper is a new way of conceptually modelling complex systems with a strong focus on their components' information flow, especially relevant for security and data protection issues and the diversity of such systems. For simplification purposes, multiple hierarchical views are created: a high-level overview of the components and their interactions of information and data as well as more detailed views for each of the sub-layers. Additionally, the interfaces between the components are defined on a high-level basis to facilitate further software architecture development.

A few strategies for analysing problem domains and organising complex systems into simplified models have been introduced, but do not offer a holistic approach for internationally co-produced software solutions, according to the problem domain defined in the Introduction in Section 1. Integrating technologies by different companies from various countries poses challenges that are more or less unique to international research projects. While the approach to modelling complex systems presented in this paper combines multiple existing strategies, the combination is uniquely adapted to suit the complexity of international software-integration projects, and account for security and privacy by design in such complex environments.

### 3 DEVELOPMENT OF A CONCEPTUAL INFORMATION FLOW MODEL

Integrating existing systems, often including legacy software, poses unique challenges. This section introduces a procedure to design information flow models based on using a soft systems thinking

approach and combining it with specific modelling techniques, allowing for the required flexibility to cover all aspects of such a conceptual model.



**Figure 1: Approach to designing a conceptual information flow model**

Figure 1 outlines each step to developing a holistic information flow model of a complex system using the proposed modelling technique. The first step is to analyse the system and its' components using Soft Systems thinking workshops, the results of which are depicted in the model structured in three layers: Extraction, Transformation and Provisioning. Each of the layers includes all components with their relevant information flows between themselves and the other layers'. Section 3.1 gives an overview of the conceptual ideas and approaches the presented modelling strategy is based on. Additionally, in Section 3.2 the entities used to visualise the conceptual model, generated by strategies introduced before, are presented.

#### 3.1 Approach to Conceptualization

When developing a system from existing software components, identification and typification of information flows is an essential task. This can be achieved by defining what information is required and the state of each of the existing tools in terms of services, technologies and data. The information flow of the system will depend on which components it comprises of and which type of data they process. Therefore the first step when conceptualizing such a complex system must be the definition of which components are to be included. Each component is then analysed thoroughly by collecting answers to the following questions:

- **Which** internal processes take place?
- **What** data do they require?
- **How** can this component communicate with the necessary input and output components?

The gathered information will influence the layout of the system as well as the interfaces between the components. These interfaces require a detailed definition as early as possible during the software development process, since this ensures all technology providers involved in the final software solution know exactly how and what to communicate with the other components. Interface definition standards are diverse and the chosen approach will be described in detail in Section 3.2.

When designing a monitoring system such as CS-AWARE, the strategy should be to first look at the monitoring tools and the services to be monitored separately, but in the end consider the system as a whole entity of a complex system. This approach will ensure a holistic analysis of the problem domain and a final model covering all particularities of the domain. For the approach presented in this paper, workshops applying the Soft Systems Methodology were chosen to offer more detailed information on the requirements to the new system. In particular for cybersecurity related software, such workshops offer insights into potential threat vectors or specific, potentially undocumented, information flows, known only subconsciously by the stakeholders of the problem domain. As described in Section 2 this type of structuring method allows the participants to create rich pictures (free-form visualisations), which creatively visualize certain aspects of a problem domain. These pictures not only allow the stakeholders to illustrate their perception to their colleagues, but also simplify the following structuring process of the SSM analyst. SSM proves to be a powerful approach, since connecting different types of stakeholders of a single problem domain typically shows how different perceptions of a system and its domain can be. The confrontation of the participants and their views allows a holistic result. Additionally, such stakeholder workshops can disclose issues of the domain that have previously been unknown to stakeholders and analysts alike by uncovering tacit knowledge of the participants.

The conceptualization of such a complex system and its information flows requires a high-level and abstract approach to include all relevant information. For the purpose of modelling a complex system of software components and their information flows, a conceptual modelling approach was considered appropriate. The approach of conceptual modelling is used in many domains and across multiple disciplines, the specific definition of which therefore varies. Especially in a security context this can include knowledge and information not initiating from SSM workshops but external experts. Ideally such domain experts would be part of the SSM sessions to guide train of thoughts of the participants in the relevant directions.

When constructing a conceptual model of a complex system the identification of the components and their information flows, as well as a holistic view of the problem domain is essential not only for architecture but also for security purposes.

The first steps of the SSM were applied during the SSM-like interviews and user workshops - *entering the problem situation*, *expressing the problem situation* and *formulating the root definitions of the systems behavior*. When beginning to model the first drafts of the information flow model the next steps of SSM were taken: *building conceptual models* and *comparing model to real-life situations*. Based on the last observations *possible changes were defined* and

the model revised accordingly. The final model is currently used as a guideline for the systems development and implementation and has therefore satisfied the last step of the SSM - *improving the problem situation* by guiding implementation decisions.

### 3.2 Modelling the Information Flow

Once a conceptual model is established, modelling the information flow is an essential aspect of understanding the interactions between components in great detail. So these methods can be applied to any new software system comprising multiple existing software tools, a strategy which will be clearly outlined in this Section. After analysing the problem domain in stakeholder workshops using SSM, the environment of the new systems and its requirements should be identifiable as well as the components to be included after a prior analysis. Additionally, SSM-like sessions with the technology partners must be scheduled, during which the specifications of each existing software, and the requirements it has with respect to the connected technology, are to be identified.

To further facilitate the development of the information flow model, three logical layers can be included, which are based on the ETL model [9] - extraction, transformation and load - a standard for database processes in data warehouses. Extraction stands for the gathering of the data from various sources, transform for cleaning and manipulation of data to ensure integrity and completeness and load for transferring the data into its target space [1]. This model was adapted by changing the load layer to data provisioning layer, since generated knowledge and information are provided to the end users in this layer. This is applicable for any software solution that provides a service based on data to their end-user. Each of the layers include components responsible for handling steps in the respective layer, structuring them and their interactions clearly. This does not only simplify further specification in the software development process but also prior communication between the technology providers.

A more detailed view of each of the layers is recommended, the concept of which is presented in Figure 2. This representation, where the other two connected layers are visualized as black boxes, shows a more detailed view of the processes occurring in the respective layer and the information flows between them. This illustration helps clarify which sub-components require what information and how this can be provided. The following images in Table 1 show the different visual modelling entities used in this conceptual modelling approach.

Information flows as shown in Figure 3 can be either of controlling, data transmitting or guiding nature - each of which has an individual graphical representation and can be either uni- or bidirectional.

Additionally, the information flows can be classified as either manual (m), automated (a) or semi-automated information flow, the latter representing flows of data that might require manual adaptations or at least the users approval before sending. Another existing model used to simplify the information flow is the I/P/O model [7] - input, process, output - and was mainly used to describe high-level interfaces between the components. First the data format and type of the input is defined, followed by the processes that take

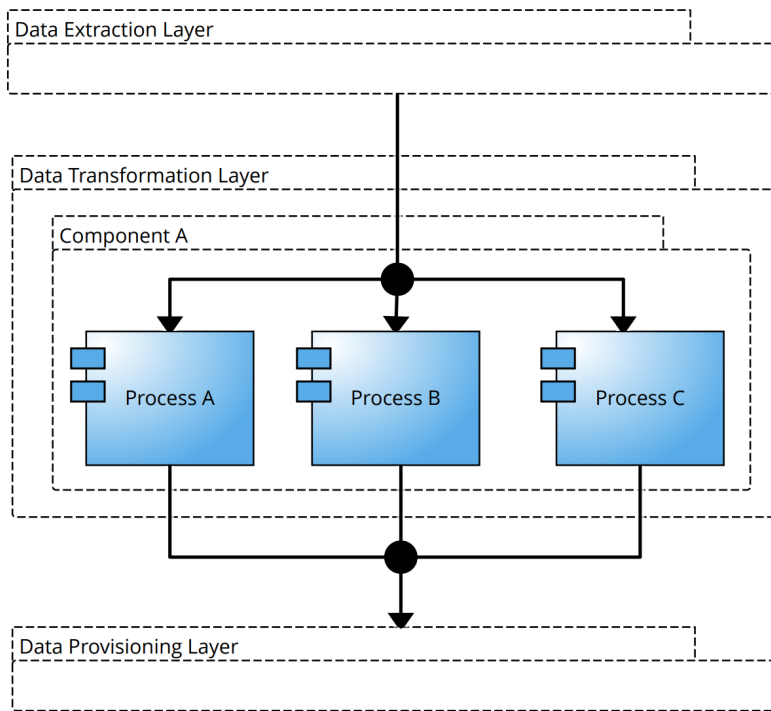






Figure 2: Example Transformation Layer

Table 1: Visual entities information flow model

Entity	Description
	An <b>external component</b> represents any data source or receiving entity that is connected to one of the internal components and shares a type of information flow as mentioned in Section 2.
	<b>Manual tasks</b> are typically analysis tasks, such as holding SSM workshops or conducting an expert analysis of certain aspects of the problem domain.
	<b>Subcomponent and component entities</b> can not only represent technologies as a whole in a high-level representation, but also their subcomponents when looking at a single layer.
	<b>Static data</b> represents data bases or configurations that are static and defined prior to implementation by experts or analysts. Since software development is an iterative process, these lists and data bases can be updated if required.

**Table 2: Interface Definition Schema**

<b>Input</b>	<b>Name of Component</b>
Source Module	Name of Component
Data Format	Data format used by the component
Description	any information required, but must include type of incoming flow
<b>Output</b>	<b>Name of Component</b>
Destination Module	Name of Component
Data Format	Data format used by the component
Description	any information required, but must include type of incoming flow
<b>Process</b>	<b>Name of Process</b>
Module/s A	Name of input components
Module/s B	Name of output components
Process	Name of process
Definition	Description of what happens in subcomponent
Data Input Format	Data format used by input components
Data Output Format	Data format used by output components

**Figure 3: Types of information flow**

place in each of the components and finally which format the output is provided in. The categories Input, Process and Output are defined for all components as can be seen in Table 2, each of which can include multiple instances of the categories.

This structure facilitates the alignment of the existing technologies immensely and clearly structures communication between the components. Also the responsibilities of data adaptations can be specified this way, aligning the data formats used throughout the system. One of the topics analysed for this model were the relations between the components and what defines them, it became clear that while all can be summarized under the term *information flow* they differ in content. The differentiation was made between data and control flow, where they first describe actual data transfer for analysis purposes and the latter conceptual and logical information which has influence on the execution of the receiving component. The last relation, that of guidance, represents a relation where one component has an extensive effect on another. While the first does not actually send data, it provides information to define the construction and/or processes of the latter.

#### 4 CASE STUDY

The case study presented in this paper depicts the information flow model for the cybersecurity awareness solution realized in the European H2020 project CS-AWARE. The project aims to improve cybersecurity in local public administrations (LPAs) by providing an online monitoring and awareness system that is able to detect security incidents by monitoring the complex organizational systems at key locations, and set it in context with information collected from external sources like cybersecurity information sharing communities or network and information security (NIS) competent

authorities, as specified by the European cybersecurity strategy [5]. This allows to classify suspicious events and incidents to concrete threats and attacks, as well as applicable strategies for prevention or mitigation. CS-AWARE can act as a decision support system by visualizing the incidents and their classifications in order to allow an administrator or system operator to take informed prevention or mitigation decisions. Additionally, CS-AWARE can act as a system self-healing solution if purely technical solutions to a specific threat or attack could be derived from the analysis of available information. Furthermore, CS-AWARE is designed to interact with cybersecurity information sharing communities to share information about newly discovered incidents that could not be classified, in order to allow the community to analyze those events and potentially help others affected by the same incident.

Figure 4 shows the high-level model of the information flow of the CS-AWARE solution. Each of the components is the responsibility of one of the projects technology partners and consists of multiple sub-processes. One of which includes the elements of the monitored LPA system as an integral part of the architecture. These are described further in more detailed models of each individual layer, divided according to the concept introduced in Figure 2, to be able to model the data transformations in each sub-component. As mentioned above, the identification of the capabilities of each component was performed during SSM-like sessions with the project partners as well as workshops with LPA partners to be able to understand the most critical information flows in the monitored systems. The results of these analyses are shown in Figure 4. The Data Extraction Layer covers the components for System Dependency Analysis, Data Collection as well as the static components of Public Information Sources and Local Public Administration Specific Internal Information Sources. The Transformation Layer includes all components responsible for adaptation and analysis of the collected data: Multi-Language Support, Data Pre-Processing and Data Analysis and Pattern Recognition. Finally, the Provisioning Layer consists of all components that provide data to users or

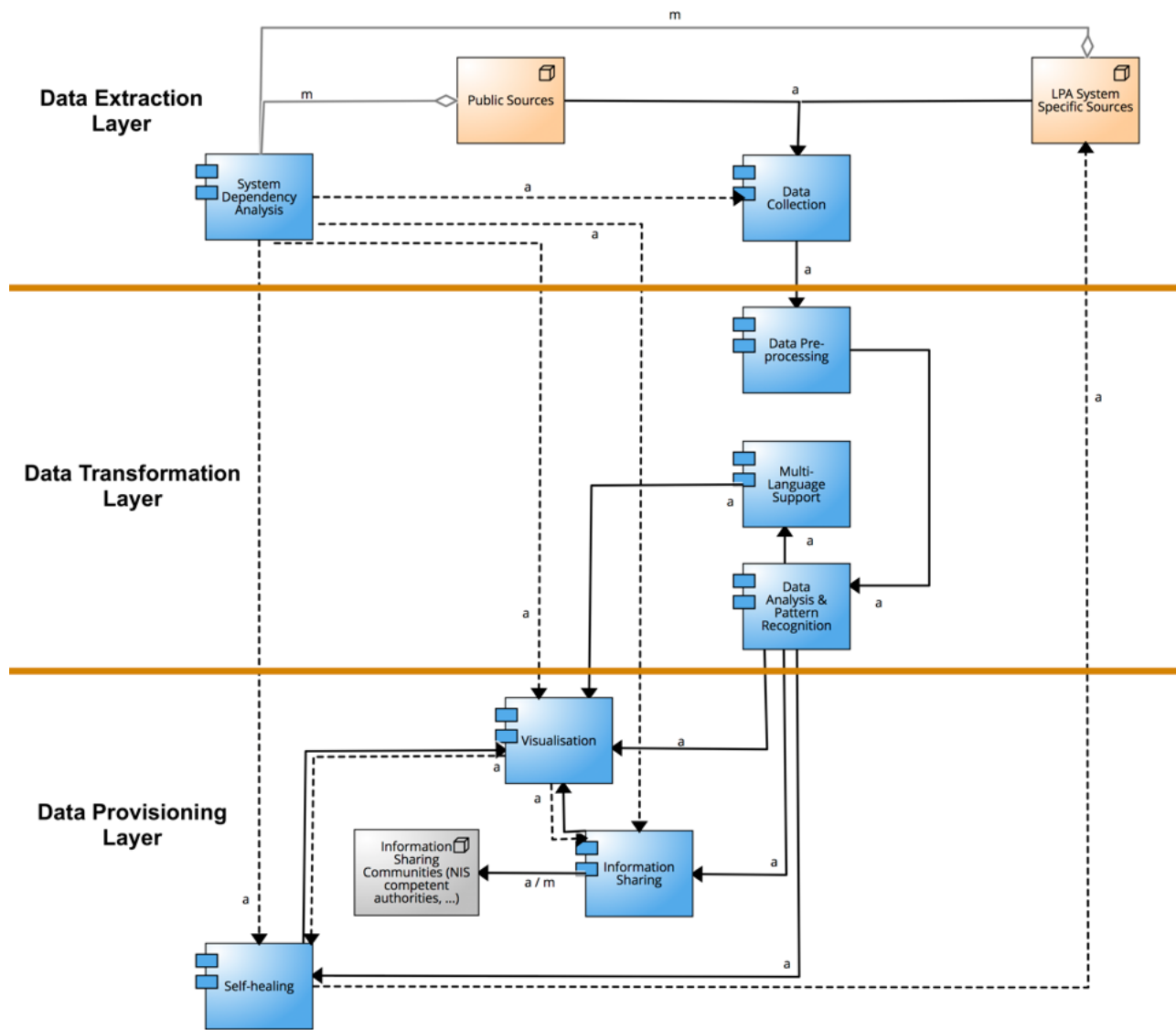


Figure 4: CS-AWARE Information Flow Model

other external entities. These are the Visualisation, Information Sharing and Self-Healing component.

The aim of the information flow model is to provide a unified understanding of which components interact with each other and in what way this interaction is made possible. The model provides a high-level overview of the main components, most of which are represented by one of the technology partners, as well as a more detailed view of the main subcomponents or processes each of them consist of. Additionally, the relations between these components are defined as well as, in the case of data flows, the data format in which the exchange takes place. The detailed software architecture is based on the conceptual information flow model and the interface

specifications. This will ensure all partners share the same understanding of responsibilities and requirements of their components during the development and implementation phase.

The design approach presented in this paper solves the problem of complex system component integration in a security sensitive setting. Proof of work is provided by the hands-on application of the method in the CS-AWARE information flow model, where multiple technologies, monitoring systems and systems to be monitored were integrated.

## 5 CONCLUSION AND FUTURE WORK

The presented approach for modelling a complex system created by merging existing software technologies offers new approaches

to structuring such models. Even though the strategy was developed based on the problem domain of CS-AWARE, it is expected to prove useful to other projects involving the fusion of existing technologies to create a new solution. Especially since we have seen the importance of having a clear structure and defined model in such complex projects. This way of designing a conceptual model of the information flow of a complex system provides a detailed basis for software adaptation and integration. Especially for international projects, the approach described in this paper allows for the consideration of cultural and organisational differences, as well as technological preconditions of the existing technologies. In addition it provides a basis for security and privacy by design, by clarifying the information flow of the system and therefore allowing a better basis for design decisions. Future work in regards to this approach will be part of the CS-AWARE project and further provide insight on the quality of the presented conceptual modelling of information flows in complex systems.

## ACKNOWLEDGEMENTS

The authors would like to thank the EU H2020 project CS-AWARE (grant number 740723) for supporting the research presented in this work.

## REFERENCES

- [1] Srividya K Bansal and Sebastian Kagemann. 2015. Integrating big data: A semantic extract-transform-load framework. *Computer* 48, 3 (2015), 42–50.
- [2] Jan Bosch. 2004. Software architecture: The next step. In *European Workshop on Software Architecture*. Springer, 194–199.
- [3] Peter Checkland. 1981. Systems thinking, systems practice. (1981).
- [4] Patrice Clemente, Jonathan Rouzaud-Cornabas, and Christian Toinard. 2010. From a generic framework for expressing integrity properties to a dynamic mac enforcement for operating systems. In *Transactions on computational science XI*. Springer, 131–161.
- [5] European Commission and High Representative of the European Union for Foreign Affairs and Security Policy. 2013. Cybersecurity Strategy of the European Union: An Open, Safe and Secure Cyberspace. JOIN(2013) 1 final. (2013).
- [6] Valentina Ferretti. 2016. From stakeholders analysis to cognitive mapping and Multi-Attribute Value Theory: An integrated approach for policy support. *European Journal of Operational Research* 253, 2 (2016), 524–541.
- [7] Jeffrey O Grady. 1995. *System engineering planning and enterprise identity*. Vol. 7. CRC Press.
- [8] JC Jiang, JY Yu, and JS Lei. 2015. Finding influential agent groups in complex multiagent software systems based on citation network analyses. *Advances in Engineering Software* 79 (2015), 57–69.
- [9] Anastasios Karagiannis, Panos Vassiliadis, and Alkis Simitsis. 2013. Scheduling strategies for efficient ETL execution. *Information Systems* 38, 6 (2013), 927–945.
- [10] Jeff Kramer. 2007. Is abstraction the key to computing? *Commun. ACM* 50, 4 (2007), 36–42.
- [11] Qing Li and Yu-Liu Chen. 2009. *Modeling and Analysis of Enterprise and Information Systems: from requirements to realization*. Springer.
- [12] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. 2008. COUNCIL DIRECTIVE 2008/114/EC of 8 December 2008 on the identification and designation of European critical infrastructures and the assessment of the need to improve their protection. Official Journal of the European Union L 345/75. (2008).
- [13] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. 2016. DIRECTIVE (EU) 2016/1148 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union. Official Journal of the European Union L 194/1. (2016).
- [14] Stewart Robinson. 2008. Conceptual modelling for simulation Part I: definition and requirements. *Journal of the operational research society* 59, 3 (2008), 278–290.
- [15] Stewart Robinson. 2008. Conceptual modelling for simulation Part II: a framework for conceptual modelling. *Journal of the Operational Research Society* 59, 3 (2008), 291–304.
- [16] Ric Roca, Dale Pace, Stewart Robinson, Andreas Tolk, and Levent Yilmaz. 2015. Paradigms for conceptual modeling. In *Proceedings of the 48th Annual Simulation Symposium*. Society for Computer Simulation International, 202–209.
- [17] Najah Ben Said, Takoua Abdellatif, Saddek Bensalem, and Marius Bozga. 2014. Model-driven information flow security for component-based systems. In *Joint European Conferences on Theory and Practice of Software*. Springer, 1–20.
- [18] Lorenza Saitta and Jean-Daniel Zucker. 2013. *Abstraction in artificial intelligence and complex systems*. Vol. 456. Springer.
- [19] John Sokolowski, Charles Turnitsa, and Saikou Diallo. 2008. A conceptual modeling method for critical infrastructure modeling. In *Simulation Symposium, 2008. ANSS 2008. 41st Annual*. IEEE, 203–211.