Supplementary information - Transparency by default: GDPR Patterns for Agile Development

A. Transparency Requirements Patterns

The methodology is driven by patterns covering three distinct aspects of the system, i.e., data, use cases, and processes, each with their own means of transparency. Each aspect is supported by a set of patterns. These patterns are motivated by backword and forward traceability. The Data Driven Patterns (DDP) include static (type-level) patterns which describe the information related to the data schema identified by classes, attributes, and associations, and dynamic (instance-level) patterns covering the actual data state, e.g., to inform the user what data is currently stored about them and when it is being used.

Use Case and Process patterns are also sub-divided into static and dynamic patterns. For example, a static process pattern provides information about a particular process in general while a dynamic one reports on its execution, including its control state or enabled actions. Additionally, both domains have alternative patterns that identify which enabled and available alternative sequence of processes or use case the system needs to reveal to the user in case the main path of execution may fail. GDPR-driven transparency patterns are special data-driven patterns derived from GDPR requirements. They include the Data Protection and Data Subject Right patterns. The first generates transparency requirements for specified data the system holds about the user falling under data protection legislation. The second pattern provides information about the users' rights on that data. We define the following patterns:

1. Data Transparency
    a. Static Data Transparency Pattern
    b. Dynamic Data Transparency Pattern
2. GDPR Driven Transparency
    a. Data Protection Transparency Pattern
    b. Data Subject Right Transparency Pattern

3. Use Case and User Story Transparency

    a. Static Use Case Transparency Pattern

    b. Dynamic Use Case Transparency Pattern

    c. Alternative Use Case Transparency Pattern

4. Process Transparency

    a. Static Process Transparency pattern

    b. Dynamic Process Transparency Pattern

    c. Alternative Process Transparency Pattern

Each transparency pattern demonstrates how to write the resulting requirement. Some patterns specify conditions when to be transparent. Conditions are logical statements composed of combinations of data operations with the actors performing them, of the form:

«actor» «data operation» «data type»

For instance, in a condition “system admin updated address”, “system admin” is the actor, “updated” the data operation and “address” the data type.

1.1 Transparency Requirements Patterns

Data Transparency

Static Data Transparency Requirements Pattern

Basic Details

Manifestation:	Standard
Belongs to domain:	Data
Pattern author:	B. Zieni

Applicability	Use to provide information about the type of data the system holds, the actions on these data types and by which actors they are performed.
Content	Data interest. Data type System actions and their actors.
Derivation of Contents	The data interest is defined by the requirements analyst based on user concerns and regulation. Data types are identified by labelling them with data interests they belong to. Checking the existing use cases that deal with this data type, a list of actions and a list of actors performing them is created.

Template(s)

Summary	Definition
Template for «data type»	*If «data type» belongs to «data interest»* *The system must communicate to the user that it holds data of «data type» on which «list of actions» can be performed by «list of actors»*

An application of the pattern results in an instantiation of the templet, for example:

Example(s)

Summary	Definition
Template for Address data	The system must communicate to the user that it holds data of type *Address* and on which *update actions* can be performed by the system administrator.

Dynamic Data Transparency Requirements Pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Data
Pattern author:	B. Zieni
Applicability	Use the data transparency requirements pattern to generate transparency requirements for the specified data instance that the system holds about the end user. This pattern gives information, during runtime, about what are the system actions being performed and by which actors.
Content	A reference to the data instance. Data Interest. Data type of this data. Data Gathered Condition
How to get the content	The reference to the data is derived from the data type. The data interest is defined by the requirements specialist based on common and individual concerns. Data type can be identified based on the data interest where the specialist comes up with the list of information identifiers then select the data types belonging to them. Information identifiers are the label descriptions that explain what kind of data is associated with this data interest. Specify that source of the data and where the data been gathered, inferred, or aggregated. This is become irrelevant when the user is the source.

Conditions are defined by the requirements specialist and get checked during runtime (see **Error! Reference source not found.**)

Template(s)

Summary	Definition
Transparency of Instance of «Data Type» because « condition ».	*//If «Data Type» is a «Data Interest»:* *In case <<condition >>, the system must communicate to the user << the «Data Type» (instance) \/&<<data gathered >> by <<actor>> | <<condition>> >>*

Example(s)

Summary	Definition
Transparency of *Instance of* "Address" because "system admin updated address".	In case "system admin updated address", the system must communicate to the user "University of Leicester, University road, Leicester, UK LE17RH" *<<gathered>> by system admin.*
Transparency of Instance of "Address" because "system admin updated address".	In case "system admin updated address", the system must communicate to the user "system admin updated address".

The following illustrates the GDPR transparency patterns: Data Protection and Data Subject Rights Transparency Patterns,

Data Protection Transparency Requirements Pattern:

Manifestation and name	Standard Data Protection Transparency Requirements Pattern

Domain	Data GDPR
Authors	Removed for blind review.
Applicability	Use Pattern to generate transparency requirements for the specified data types of the data falling under data protection legislation the system holds about the end user. This pattern illustrates the data accessibility and storage.
Contents	Data interest. Data types. Data storage. Data access.
Derivation of Contents	The data interest is defined by the requirements analyst based on user concerns and regulation. Data types are identified by labelling with the data interests they belong to. How is data stored in the software. Is it (encrypted (yes/no), anonymized (yes/no), Pseudonymization (yes/no)? Also, specify for how long it will be stored? Explain why if needed. Where is it stored? Specify if the data storage place, while still complying with security requirements. Specify with whom it is shared and who has access to it.

Summary	**Definition**
Template for «data type»	If «Data Type» is a «Data Interest»: The system must communicate to the user that it holds data of the type «Data Type» which has been «Data access» \/&« Data storage».

Summary	**Output**
Template for Address data	The system must communicate to the user that it holds data of the type “address” which has been “accessed by system admin.” This data is encrypted and will be stored until your account deletion.

Data Subject Rights Transparency Pattern:

Manifestation and name	Standard Data Subject Rights Transparency Pattern
Domain	Data GDPR
Authors	Removed for blind review.
Applicability	Use pattern to provide information about what data system holds about the user and their rights on that data under GDPR.
Contents	Data interest. Data types. Data subject’s rights.
Derivation of Contents	The data interest is defined by the requirements analyst based on user concerns and regulation. Data types are identified by labelling with the data interests they belong to.

	Data types are called categories of data under GDPR and specify the data subject's rights on the data. Those rights are to be informed, access, rectify, erasure, restrict processing, data portability, object, rights in relation to automated decision making and profiling. This process is performed by the requirements analyst.

Summary	**Definition**
Template for «data type»	If «data type» is a «data interest» the system must communicate to the user that it holds data of «data type» and «data subject's rights» can be performed by data subject.

Summary	**Output**
Template for Address data	The system must communicate to the user that it holds data of the type of Address and actions of erasure, restriction of processing, and data portability can be performed by the user (data subject).

B. Applying the Methodology

The methodology consistent of four steps preceded by the preliminary step of identifying data interests. Based on these, the first step applies the transparency patterns. The second step focuses on the goals that are out of scope for the current implementation of the system and the foreseeable future. The rationale behind this step is to manage the users' expectations, making them aware of limitations in relation to their goals. The third step is a consistency check for the resulting requirements.

Step 0: Identify Data Interests.

Input: User data concerns based on the goals, laws and regulation.

Output: List of data interests.

To ensure consistency of requirements, it is important to identify all the data interests prior to applying the patterns, so that the same data interest used in different areas of the application can be grouped and used in the same way. Data interests are a reference to data that are defined under the same concern or satisfying the same use goal, e.g., persona data, performance data, account data, etc. This process provides the flexibility to identify any data group related to the user. The process of generating data interests are as follows:

1. The analyst identifies data in the system by a name, e.g. “personal data” as defined under GDPR [11].

2. The specialist selects the data types that are mapped to the data interests. This mapping comes from selecting the relevant data classes that represent the data from the conceptual data model.

For example, users concern about data is often shown in their goals e.g. “I want to see my shipping address before confirming payments”. A n analyst in this case creates a data interest called “shipping data” to represent this user data concern.

Step1: Apply transparency patterns to implemented goals.

Input: List of user goals that are (to be) implemented.

Output: Complete transparency requirements document (TRD).

We map the goals to one or more of the pattern categories (Data, GDPR, Use Case, Process). Goals can refer to functionality already in the system, or to be added in the next iteration. Based on the use cases or user stories selected we use the pattern prompt list to choose the corresponding patterns. Prompt list guides the stakeholders to map use case/user story with the corresponding domain driven pattern by describing common features of a particular DDP.

Then for each data interest from step 0, identify the corresponding data patterns and generate the requirements by following the pattern template which are then added to the transparency requirement documentation.

For example, from a user story such as “As a user, I want to make an online purchase using quick checkout and have it shipped to my current address, so that I can order as fast and hassle free as possible”, our patterns will generate the following requirements:

GDPR Data Protection pattern: The system must communicate to the user that it holds data of type Address which has been “accessed by system admin” and “This data is encrypted and will be stored until your account deletion”.

Static Use case pattern: The system must communicate to the user that “default address must be indicated before quick checkout”.

Step 2: Add justifications for unimplemented goals.

Input: List of all unimplemented goals (e.g., from development backlog)

Output: List of transparency requirements relating to unimplemented goals.

We assign each goal one of the following justifications: WNS = Why Not in System, RWS = Relation with System, ESB = Explain System Boundary, GAG = Give Alternative Goal and record a reason why that justification has been assigned. The goals are then clustered by reason to create a single transparency requirement to inform the user why a particular goal has not been implemented. This gives the user a better understanding of the limitations of the system and improves their mental model. As these goals are not in the focus of the system, transparency about them is usually achieved through “pull transparency” where the user has to actively call for the information.

For example, in food delivery systems a user goal can be “I want to order my groceries”. The justification here would be ESB where the system currently only delivers fast food, because both fast food and groceries are food types.

Step 3: Consistency check

Input: TRD (from step2).

Output: Finalized TRD.

We finalise the transparency requirements document by applying a consistency check that results in removing duplicates and conflicting requirements, as well as aggregating requirements that can be combined. The requirements analyst needs to check if there is any information disclosure or conflict with the other requirements, in particular in security or privacy.

For example, sharing information about data servers could cause a security breach.