Supplementary Information_TEM

1.1 Overview of the Methodology

The engineering methodology is performed during the software development process. The methodology is part of the requirements engineering process, which can be organised in different ways according to the process model applied.

The steps and patterns are applied by requirements specialist and developers. The requirements specialist generates the transparency requirements using the transparency patterns. The methodology helps the requirements specialist to decide about the presentation choices when passing information to the user through first separating the content from the design options, then guiding the specialist using transparency presentation concepts that explain how to pass the information to the user. The methodology is structured in a way so that the different patterns are decoupled and can be applied in different locations and at different times, by different teams of experts.

The requirements specialist responsibilities are assigned for various roles depending on the development process type. For example, in the agile processes the product owner is the one who is the most heavily involved with the requirement practices, then the SCRUM master and SCRUM team are mainly involved in the requirement analysis processes.

The methodology consistent of four steps. These steps are showing in the conceptual framework see **Error! Reference source not found.**. The first step applies the transparency patterns. The patterns capture transparency concerns using user stories, use cases and processes, as well as based on data interests. The second step focuses on the goals that are out of the scope for the current implementation of the system (and for the foreseeable future). The rationale behind this step is to not violate the users' expectations about the system by making the user aware of the limitations of the system in relation to their goals. The third step is consistency and conflict check for the high-level resulting requirements. The steps are demonstrated in Figure.1. In this methodology the developers can use user stories instead of the use cases. For that to be possible the developers need to make sure that all the components are available in their user stories, for instance the pre- and post-conditions, the main and alternative sequences.

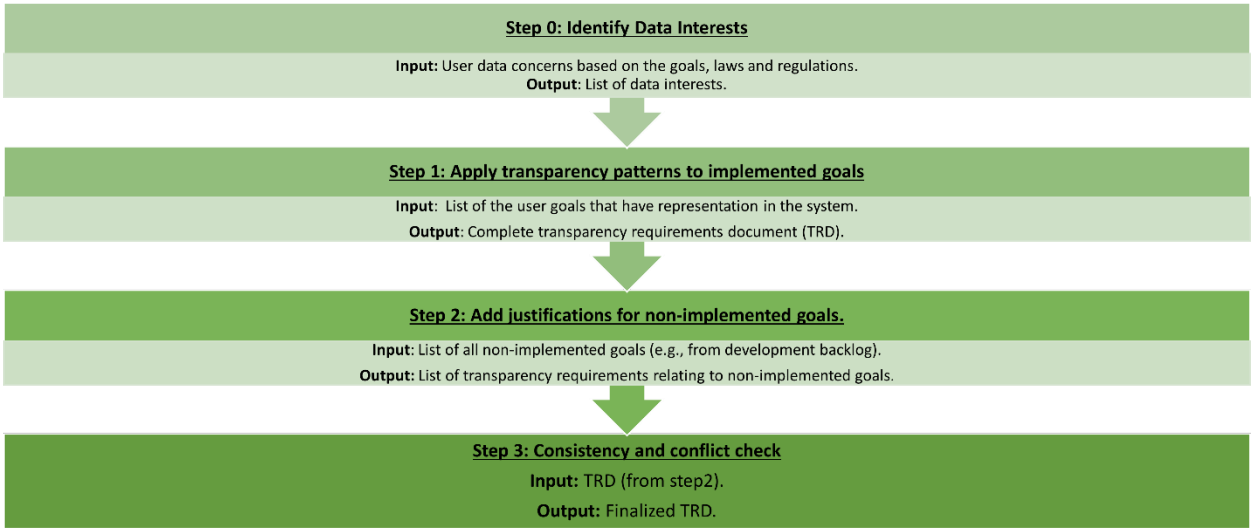

Figure.1 TEM steps

1.2 Methodology Steps

Step 0: Identify Data Interests.

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

Action: To ensure consistency of requirements, it is important to identify 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 is described in Section **Error! Reference source not found.**

Output: List of data interests.

For example, users' concern about data is often shown in their goals e.g. "I want to see my shipping address before confirming payments". An engineer in this case creates a data interest called "shipping data" to represent this user data concern.

Step 1: Apply transparency patterns to implemented goals (Use Cases, User stories)

Input: List of the user goals that have representation in the system.

Action: Map the goals to one or more of the domain driven patterns (Data, Use case, Process), this can be done directly or with the aid of the prompt list. The goals can refer to functionality already

in the system, functionality overlooked by the user or to functionality to be added to the system. In order to achieve this, the implementer must list the relevant use case or user stories representation in the software. Then use the pattern prompt list to identify the related components of system use cases to help you to choose the corresponding patterns. Use the data interest identification process to generate the data interests. For each data interest identify the corresponding data patterns. Write the resulting requirements in the Transparency requirements document (TRD). See a full description on Transparency patterns in **Error! Reference source not found.**.

To increase the consistency over the application, it is important to identify all the data interests within the team hierarchy, where the project managers/domain experts can define and then share them with the other team members.

Output: Complete transparency requirements document (TRD).

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 is "accessed by delivery service" and "This data is encrypted and will be stored until your account is deleted".

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

Step 2: Add justifications for non-implemented goals.

Input: List of all non-implemented goals (e.g., from development backlog).

Action: Create a list or, use existing list of the goals for that purpose. The low priority goals are to be implemented later or considered out of scope for the foreseeable future. After that, select the related goals which are the ones that could fit into the system scope, but they are currently not implemented.

Then run the following Justifications on the mentioned goals, (WNS = Why Not in System, RWS = Relation with System, ESB = Explain System Boundary, GAG = Give Alternative Goal). 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 must actively call for the information.

Output: List of transparency requirements relating to non-implemented goals.

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 and conflict check

Input: TRD (from step2).

Action: Streamline the transparency requirements document by applying a consistency and conflict check that results in removing duplicates and conflicted requirements as well as aggregating requirements that can be combined. The requirements specialist needs to check if there is any information disclosure or conflict with the other requirements like security or privacy.

Output: Finalized TRD.

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

1.3 . 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.

1.3.1 Data Transparency

Static Data Transparency Requirements Pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Data
Pattern author:	B. Zieni
Applicability	Use the static data type transparency requirements pattern to provide information about the data the system holds about the user and what are the existing system actions on that data type by which actors.
Content	Data interest. Data type System actions and their actors.
How to get the content	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.

By checking all the existing use cases that deal with this data type, a list of operations and a list of actors performing them can be identified. The identification process is done by the requirements specialist based on the data interest.

Template(s)

Summary	Definition
Transparency about «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*» « list of *data operators*» can be performed by « list of *actors*» &/<<*Reasons*>>

Example(s)

Summary	Definition
Transparency about "Address"	The system must communicate to the user that it holds data of the type "*Address"* and "*update"* can be performed by "System admin"

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.

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".

1.3.2 GDPR Transparency

Data Protection Transparency Requirements Pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Data

Pattern author:	B. Zieni
Applicability	Use the Data Protection Transparency 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.
Content	Data Interest Data type Data storage Data access
How to get the content	The data interest is defined by the requirements specialist based on common and individual users' 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. Data type is called category of data under GDPR. How is data is 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.

Template(s)

Summary	Definition
Transparency about data type «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 >>|*

Example(s)

Summary	Definition
Transparency about data type “Address” because system admin updated address.	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 Right Transparency Requirements pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Data
Pattern author:	B. Zieni
Applicability	Use data subject right transparency requirements pattern to provide information about what are data types that system holds about the end user and their rights on that data under GDPR
Content	Data interest.

	Data type Data subject's rights
How to get the content	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. Data type called category of data under GDPR. Specifying the data subject's rights on the data. They are to (be informed, access, rectify (complete the incomplete data), erasure, restrict processing, data portability, object, rights in relation to automated decision making and profiling.)

Template(s)

Summary	Definition
Transparency about «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*» and << Data subject's rights >> can be performed by data subject

Summary	Definition
Transparency about "Address"	The system must communicate to the user that it holds data of the type "*Address"* and "erasure, restrict processing, data portability*"* can be performed by data subject.

1.3.3 Use Case Transparency

Static Use Case Transparency Requirements pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Use case
Pattern author:	B. Zieni
Applicability	Use the Static Use Case Transparency Requirements pattern to provide information about how the use cases of the system work. The requirements specialist makes a choice on when to use this pattern either when it is a new use case or a use case that needs more information.
Content	Use case Pre/post conditions for use case actions Main sequence
How to get the content	The use case from the use case diagrams the specialist chooses to be transparent about. Pre/post conditions are part of use case description. Both workflows and use case scenarios may define alternative sequences of actions. Main sequence from the use case diagram.

Template(s)

Summary	Definition
Transparency about «Use Case»	*The system must communicate to the user that <<about pre-condition |/& about post-condition |/& follow main sequence >> <<before |/& after |/& to perform>> «Use Case».*

Example(s)

Summary	Definition
Transparency about "withdraw money"	The system must communicate to the user that "your balance should be above the overdraft" before "withdraw money".

Alternative Use Case Transparency Requirements Pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Use case
Pattern author:	B. Zieni
Applicability	Use pattern to identify which enabled and available use cases the system needs to reveal to the user in case of a failure to meet pre/ postconditions.
Content	Pre/post conditions for use case actions. Use case. Alternative use cases. Reason describes why the system has chosen this sequence.

How to get the content

Pre/post conditions are part of use case description. Both workflows and use case scenarios may define alternative sequences of actions.

Use case that has alternative sequence from use case diagram.

The list of alternatives (available and enable) use-cases for the use case that match the current system state or ways to fulfil the pre and/or postconditions from the use case diagram.

Reason from the use case diagram illustrated from the deviation and mismatch of the pre and postcondition.

Template(s)

Summary	Definition
Transparency about «Use Case» because of unmet <<pre-conditions | post-conditions>>.	*If current system state doesn't meet <<pre-conditions "at start" | post-conditions "at end" >> of <<use case>> the system must communicate <<list of alternative available and enable use-cases for <<use case>> that match the current system state or ways to fulfil the <<pre &| post conditions >>>> because of <<reason>>*

Example(s)

Summary	Definition
Transparency about "transfer money to account at same bank" because of unmet	If current system state doesn't meet "enough funds in own account" at start of "transfer money to account at same bank" the system must communicate "deposit money in bank

"enough funds in own account".	account does not own by user" because of "not enough funds are less than the minimum amount of 150 GBP".

Dynamic Use Case Transparency Requirements Pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Use case
Pattern author:	B. Zieni
Applicability	Use the Alternative Use Case transparency requirements pattern to identify, which alternative sequence of use cases the system needs to reveal to the user.
Content	System use case that has alternative sequences that the system chosen**.** Alternative sequence.
How to get the content	Use case that has alternative sequences from the use case diagram, where the system is an actor. The list of alternative sequence of use-case from the use case diagram.

Template(s)

Summary	Definition
Transparency about «Use Case» because of alternative sequence.	*If the system uses an alternative sequence of << use case>>. The system must communicate <<alternative sequence >>*

Example(s)

Summary	Definition
Transparency about "withdraw money" because of alternative sequence.	If the system uses an alternative sequence of "withdraw money" The system must communicate "change the amount or add funds to your account"

1.3.4 Process Transparency

Static Process Transparency Requirements pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Process
Pattern author:	B. Zieni
Applicability	Use the static process transparency requirements pattern to provide information about how the processes of the system work.
Content	Process Pre/post conditions for process actions Main flow
How to get the content	Any process from the activity diagram. Pre/post conditions are part of process description. Both workflows and use case scenarios may define alternative sequences of actions. Main flow from the activity diagram.

Template(s)

Summary	Definition
Transparency about «Process»	*The system must communicate <<about pre-condition |/& about post-condition |/& follow main flow >> <<before |/& after |/& to>> «Process» to the user.*

Example(s)

Summary	Definition
Transparency about «*Withdraw money from another's bank ATM*»	The system must communicate "Your bank needs to be in the List off Banks we deal with for free charge" before «*Withdraw money from not-my-bank ATM*» to the user.
Transparency about <<transaction fee >>	The system must communicate "this product with and without transaction value"

Alternative Process Transparency Requirements Pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Process
Pattern author:	B. Zieni
Applicability	In case of a failure to meet pre/ postconditions, use pattern to identify which enabled and available alternative processes (sequences of actions) the system must reveal to the user.

Content	Pre/post conditions for process actions Process. Alternative flow Processes**.** Reason.
How to get the content	Pre/post conditions are part of process description. Both workflows and use case scenarios may define alternative sequences of actions. Process that has alternative flow of processes from activity diagram**.** The list of alternatives (available and enable) flows for the process that match the current system state or ways to fulfill the pre and/or post conditions from the activity diagram. Reason describes the failure of pre/post condition.

Template(s)

Summary	Definition
Transparency about «Process» because of unmet <<pre-conditions | post-conditions>>	*If current system state doesn't meet <<pre-conditions | post-conditions>> at <<start |end >> of <<process>> the system must communicate <<list of alternative available and enable processes for <<process>> that match the current system state>> to the user or how to fulfill the pre and/or post conditions >> because of <<reason>>*

Example(s)

Summary	Definition
Transparency about "*assurance claim*" because of unmet "*minimum contract duration*"	If current system state does not meet "*minimum contract duration at start*" of "insurance claim" the system must communicate "*call our representative to discuss your options*" because claims can only be made if the contract activation period is one year, and yours has been active for 7 months.

Dynamic Process Transparency Requirements Pattern

Basic Details

Pattern manifestation:	Standard
Belongs to domain:	Process
Pattern author:	B. Zieni
Applicability	Use the Alternative process Transparency Requirements pattern to identify, which enabled and available alternative sequence of processes the system needs to reveal to the user.
Content	Process with alternative flows. List of alternative sequences.
How to get the content	Processes can be higher-level workflows over use cases or detailed use case scenarios. The list of alternatives (available and enable) flows for the process that match the current system state or ways to fulfil the pre and/or post conditions.

Template(s)

Summary	Definition
Transparency about « process», because of alternative flow.	*If the system uses an alternative flow of << process>>. The system must communicate <<alternative flow >>,*

Example(s)

Summary	Definition
Transparency about "*withdraw money*" because of alternative flow	If the system offers an alternative flow of "*withdraw money*" The system must communicate "change the account" to the user,