Co-inform

Context Matters,
Your Sources Too

Policies and Procedures for Interventions Deployment and Testing

D2.2

#ThinkCheckShare

Document Summary Information

Project Title:	Co-Inform: Co-Creating Misinformation-Resilient Societies		
Project Acronym:	Co-Inform	**Proposal Number:**	770302
Type of Action:	RIA (Research and Innovation Action)		
Start Date:	01/04/2018	**Duration:**	36 months
Project URL:	http://coinform.eu/		
Deliverable:	D2.2 Policies and Procedures for Interventions Deployment and Testing		
Version:	V1.3		
Work Package:	WP2		
Submission date:	30/09/2019		
Nature:	P	**Dissemination Level:**	P
Lead Beneficiary:	University of Koblenz-Landau (UKOB)		
Author(s):	Ipek Baris (UKOB) Akram Sadat Hosseini (UKOB) Oul Han (UKOB) Prof. Dr. Steffen Staab (Coordinator, UKOB)		
Contributions from:	Martino Mensio (OU) Ronald Denaux (ESI) Mattias Svahn(SU) Love Ekenberg (IIASA) Eleni A. Kyza, Evangelos Karapanos, Loukas Konstantinou (CUT) Syed Iftkhar H.Shah, Nancy Routzouni, Deligiannis Athanasios, Dr. Ioannis Magnisalis (IHU) Orna Young, Allan Leonard (FactCheckNI)		

The Co-inform project is co-funded by Horizon 2020 – the Framework Programme for Research and Innovation (2014-2020) H2020-SC6-CO-CREATION-2016-2017 (CO-CREATION FOR GROWTH AND INCLUSION).

Revision History

Version	Date	Change editor	Description
0.1	8.9.2019	UKOB	Initial draft
0.2	16.9.2019	UKOB	Added introduction, policies
0.3	16.9.2019	OU	Added description for credibility module
0.4	17.9.2019	ESI	Added description for semantic analysis module
0.5	17.9.2019	UKOB	Updated policies, added table for 2nd section
0.6	17.9.2019	SU	SU workshop plan
0.6	17.9.2019	CUT	Review 0.6
0.7	17.9.2019	UKOB	Added policies in machine readable format
0.7	19.9.2019	IHU	Review
0.8	20.9.2019	IHU	Added Greek 2nd Workshop Plan
0.9	23.9.2019	UKOB	Updated introduction
1.0	24.9.2019	FactCheckNI	Review, added contents for the forms
1.1	24.9.2019	IIASA	Added content for Austria workshop
1.2	24.9.2019	CUT	Added policy evaluation technique
1.3	25.9.2019	UKOB	Edited co-creation section, review

Disclaimer

Copyright Message

Executive Summary

In this document, we describe the procedures associated with recommended platform policies and interventions. Co-Inform platform policies integrate stakeholder requirements (i.e. citizens, journalists, and policy makers) identified by WP1 and WP5, and orchestrate services developed within WP3 and WP4.

In **Section 1** we first outline how this work fits the objectives of Co-inform and more specifically of WP2. This deliverable describes how to manage the responses of the platform with user feedback, by considering user requirements from WP1 and WP5, and also the services developed within WP3 and WP4. To this aim, we describe the updated component of the policy framework (D2.1), which is an open-source ECA (Event-Condition-Action) rule engine that has been designed for handling real-time misinformation.

The policies and rules automated by the rule engine will be integrated with Co-inform tools that will be defined by WP3 and WP4. **Section 2** describes the co-creation of Co-Inform policies that will be tested by stakeholders in the second co-creation workshop. **Section 3** describes specifications for the second co-creation workshop that will evaluate the Co-inform tool prototypes provided by WP3 and WP4 and the associated policies and rules provided by WP2.

In **Section 4,** we present details of recommended policies that are defined based on user requirements obtained by WP5 and WP1 and provide human-readable formalisations and their risks.

In **Section 5**, we briefly describe the deployment of policies to the Co-Inform platform. Finally, we provide the machine-readable formalisation of each policy in the **Appendix**.

Table of Contents

Introduction 8
1.1. Objective of WP2 and Task 2.2 9
2. Co-Creation of Co-Inform Policies 10
3. Evaluation of Policies 11
4. Co-inform Policies 12
4.1. Flagging Policy 12
i. Flagging Policy #1: Showing the most critical score above threshold 13
ii. Flagging Policy #2: See more, see less button 14
iii. Flagging Policy #3: Notify user 14
4.2. Marking Policy 15
i. Blurring the whole post 15
ii. Blurring the whole post when the cursor is on top of the post 15
iii. Blurring the post partially 16
iv. Blurring the retweet button 16
v. Putting a visual stop sign on top 17
vi. Putting circle information icon on the top 17
vii. Removing a post 18
4.3. Reporting Policy 18
i. Reporting not flagged post 18
ii. Reporting flagged post 20
4.4. User Management Policy 21
i. Spammer user's policy 21
5. Deployment of Policies 22
6. Conclusion 23
7. References 24
Appendix 25
Flagging Policy 25
i. Flagging Policy #1: Showing the most critical score above threshold 25
ii. Flagging Policy #2: See more/See less 26
iii. Flagging Policy #3: Notify User 26
Marking Policy 26
i. Marking Policy #1 and #2: Blurring post full/partially 26
ii. Marking Policy #3: Blurring retweet button 26
iii. Marking Policy #4: Putting a visual stop sign 27
iv. Marking Policy #5: Putting a circle information icon on the top 27

v. Marking Policy #6: Removing a post 27
Reporting Policy 27
i. Reporting Policy #1: Reporting not flagged post 27
ii. Reporting Policy #2: Reporting flagged post 27
User Management Policy 28
i. Spammer users policy 28

Abbreviations

API	Application Programmer Interface, defines how programmers can use a service or library
JSON	JavaScript Object Notation
MCDA	Multi-Criteria Decision Analysis
N/A	Not available

List of Tables

Table 1. Flagging Policy #1 ECA formalisation in natural language 14
Table 2. Flagging Policy #2 ECA formalisation in natural language 15
Table 3. Flagging Policy #3 ECA formalisation in natural language 16
Table 4. Marking Policy #1 ECA formalisation in natural language 17
Table 5. Marking Policy #2 ECA formalisation in natural language 17
Table 6. Marking Policy #3 ECA formalisation in natural language 18
Table 7. Marking Policy #4 ECA formalisation in natural language 18
Table 8. Marking Policy #5 ECA formalisation in natural language 19
Table 9. Marking Policy #6 ECA formalisation in natural language 19
Table 10. Marking Policy #7 ECA formalisation in natural language 20
Table 11. Reporting Policy#1 ECA formalisation in natural language 21
Table 12. Form for flagging the post 21
Table 13. Reporting Policy #2 ECA formalisation in natural language 23
Table 14. Form for unflagging posts 23
Table 15. User Management Policy #1 ECA formalisation in natural language 24

Introduction

Germany's NetzDG law implementation[1] in 2017 attributed liability to platforms if they fail to remove defamatory content and hate speech within a time limit and imposed large fines up to 50 million Euro. Approaches like these that choose "the stick" over "the carrot" are currently considered to be the most effective measures for increasing (self-)policing by the big platforms.[2] The limitation is that governments merely regulate misinformation-handling policies rather than co-creating them. As a result, regulation provides quantified indices that lead to deleting more misinformation than before, but platform policies are still created and tested by the platforms, whose intention is not governance in the public interest. This is why most directives on misinformation policies ask for transparency from online platforms.

In platform policies for combating misinformation, transparency is crucial in order to balance the strengths and weaknesses of automated AI. On the one hand, algorithms efficiently automate the detection and deletion of potential misinformation. On the other hand, algorithms are limited in accuracy and endanger the freedom of expression. In order to balance the two, the fight against misinformation requires a system for integrating automated AI, real-time intervention of experts, and user input in policymaking for and by online platforms. In this spirit, the European Commission suggested platform policies in September 2017 that respond to misinformation by adjusting the ratio of automation versus involving external actors. Example policies are:[3]

- Fully automated removal should be applied where the circumstances leave little doubt about the illegality of the material
- In a limited number of cases, platforms may remove content notified by trusted flaggers without verifying legality themselves

In order to combat misinformation effectively, platform policies should increasingly involve stakeholders as part of platform policies against misinformation. Stakeholders are general users but also journalists, fact-checkers, and policymakers. Then, the Co-Inform platform must connect all stakeholder requirements with the platform functionalities. To this aim, the stakeholder requirements must be converted to the rule-based and machine-readable form of events, conditions, and actions (ECA) (as described in detail D2.1).
We analysed the feedback from the 1st co-creation workshop and from the reviewers of the preceding deliverables that have given shape to the Co-inform functionalities. From the WP5

[1] Act to Improve Enforcement of the Law in Social Networks (Network Enforcement Act, NetzDG) - Basic Information. *Federal Ministry of Justice and Consumer Protection*. Retrieved from
https://www.bmjv.de/DE/Themen/FokusThemen/NetzDG/NetzDG_EN_node.html

[2] Schiffrin, A. (2019 August). Startups and the Fight against Online Disinformation. *The German Marshall Fund of the United States*. Retrieved from
http://www.gmfus.org/sites/default/files/publications/pdf/Schiffrin%20disinformation%20startups%20-%2029%20Aug.pdf

[3] EC Communication on Tackling Illegal Content Online: Towards an Enhanced Responsibility of Online Platforms (COM(2017) 555 final). *European Commission*. Retrieved from
https://ec.europa.eu/transparency/regdoc/rep/1/2017/EN/COM-2017-555-F1-EN-MAIN-PART-1.PDF

analyses, we have yielded requirements from citizens, journalists, fact checkers, and policy-makers for combating misinformation on social media. With their requirements in mind, Co-inform policies and interventions were integrated to support each of these user groups.

1.1. Objective of WP2 and Task 2.2

WP2 aims for the flexible orchestration of services developed within WP3 and WP4 with user requirements from WP5 and WP1. In this deliverable “Policies and Procedures for Interventions Deployment and Testing”, we describe the procedures associated with platform policies and interventions. Throughout, our usage of the term “platform policies” stands for internal regulations of the final Co-inform platform that will provide users with relevant functionalities.

2. Co-Creation of Co-Inform Policies

The scope of the second co-creation workshop is to test and to give feedback about the initial version of the Co-Inform platform along with the policies (cf. D1.2)[4]. Participants of the 2nd workshop i.e. stakeholder groups (citizens, fact-checkers/journalists, and policymakers) will test and examine the tools and associated policies.

Co-creation workshops will have the following main activities:

- Organizers will present the workshop and its objectives as well as the goals of the workshops and the agenda. The organizers provide a list of attributes and characteristics of the policy recommendations developed within Co-Inform, where after the various features are discussed. Participants may also suggest further options, and extensions.
- The participants have an opportunity to discuss the options and provide feedback.
- The participants discuss more thoroughly the desired features of the policies and possible variations thereof. All prevailing and possible policy recommendations are discussed, followed by a discussion of positive and negative sides of the different options. A result of this session is a list of features that are important for the participants. These features are the criteria for the subsequent evaluation described in Section Co-Creation of Co-Inform Policies.
- The participants rank the policy recommendations under each criteria (feature) and suggest other policies or variations of the prevailing ones. The participants write down the names of all this on coloured cards, put these on a flipchart and explain their choices.
- The criteria are ranked in the same way as the tools. Each criterion is first discussed to make sure that participants agree on their definitions. The participants may also change and add further criteria.

[4] 5.4 Second Workshop: Feedback on prototype and policies
As mentioned above, the first workshop is expected to provide insights about the function of the tools and the features they are expecting to get. Following these, the second workshop is expected to test and give feedback about the generic version of the tool- that at this time - has to be provided as a prototype. They also may co-create some parts of the tools (user-interface, for instance). Stakeholders will also give feedback on strategies that seem most promising in raising awareness and addressing misinformation

3. Evaluation of Policies

For the evaluations herein, we will put the policies to the test in a series of workshops in three countries. WP1 has the task to organize these. At these workshops we will utilise the MCDA-method with a software for integrated multi-attribute evaluation under risk, subject to incomplete or imperfect information. The software originates from our earlier work on evaluating decision situations using imprecise utilities, probabilities and weights, as well as qualitative estimates between these components derived from convex sets of weight, utility and probability measures. During the process, we consider the entire range of values as the alternatives presented across all criteria as well as how plausible it is that an alternative outranked the remaining ones, and thus provide a robustness measure. We will use the state-of-the-art multi-criteria software DecideIT[5], which allows for imprecision of the kinds that exist here.

At the workshops we will also employ Repertory Grid Technique (Curtis, Wells, Lowry, & Higbee, 2008).This technique consists of two parts: the semi-structured interview which takes place first and the rating exercise. Firstly, the nudging interventions are presented to the participants in sets of three (i.e., triad) and asked to "identify a way in which two of the concepts are alike and different from the third", resulting to the elicitation of personal opposite constructs. These constructs will be used later on, when every participant needs to rate all concepts individually, against her own elicited personal constructs, by using scales. The output of these procedures will undergo qualitative (i.e. content analysis) and quantitative data analysis with DecideIT and Nvivo[6].

Policies which we will describe in Section 4 will be in the form of platform settings or prototypes with different behaviors. In the second workshops, the users will test them and give feedback about each policy. Those policies which might not be deployed by the platform will be evaluated with flashcards. We additionally ask their opinions on alternative solutions in form of ECA (Bae. Bae, Kang & Kim, 2004) (i.e. when this event happens, and under these conditions, what would you expect to see from the platform, how would you react to it). In this way, we will collect new policy recommendations.

[5] https://www.preference.nu/decideit/
[6] https://www.qsrinternational.com/nvivo/nvivo-products

4. Co-inform Policies

Co-Inform policies orchestrate the technological tools that are developed within the project. The goals of Co-Inform policies are to:

- deal with critical misinforming posts
- reduce the workloads of WP4 modules (e.g. plugin, dashboard, etc.)
- improve the functional modules of WP3 with detailed user feedback which will be assessed in WP5
- manage the Co-Inform users for preventing abuse of the platform.

4.1. Flagging Policy

The Co-Inform platform offers technological tools such as a browser plug-in and a dashboard. These tools are integrated with the modules to be developed within WP3. The modules produce scores for identifying misinformation as follows:

- **Similarity analysis** produces a score of similarity between other misinforming posts
- **Semantic analysis** is a low-level service which produces various features which can be used to build credibility predictors for content that cannot be directly linked to human-produced credibility scores.
    - It extracts main topics, emotions, entities, words and phrases (described in detail in D3.1) and relevant sentences from textual content (a post, a user timeline or an article accessible on-line).
    - These features can be used in various ways (but require building custom higher-level services), e.g.:
        - They can constrain the scope of policy rules based on a particular combination of topics/entities (e.g. categories: Politics, Lobbying; entity: "Migration Watch UK")
        - they can be used as features to a machine learning model to produce a credibility prediction (based on some relevant dataset)
        - they can be used to find/recommend related articles
        - they can be used to analyse the spread of claims/topics
- **User tagging** shows the number of users that tagged the post.
- **Credibility analysis** produces a score of credibility to both users (e.g. twitter profiles) and posts. The credibility is computed by matching the entities linked in the posts (URLs) with the historical data coming from fact-checkers. A post can be misinforming because it contains a URL that has been fact-checked or because it comes from a news source that has been related to misinformation in the past.
- **Content analysis** provides two main scores. The first one is a score of stances towards the claim-in-discussion. Stance is characterized as *supporting* the claim, *querying* the claim, *not related* to claim, and *denying* the claim. The module gives a probabilistic estimation of each group. The second one is a veracity estimation of the claim-in-discussion by aggregating information from the stance analysis. It gives an estimation of how the post's veracity is likely to be false.

The goal of the *flagging* policy is to deal with misinforming posts. We recommend the following flagging policies that will be tested at future co-creation workshops:

i. Flagging Policy #1: Showing the most critical score above threshold

When the system receives scores from each module, and if the scores are above threshold of their module, the system shows those scores to the users.
The system allows the user to change the threshold for each module. If user defines a custom threshold for the modules, the system filters critical scores based on user-defined thresholds.

ECA formalisation in natural language:

Table 1. Flagging Policy #1 ECA formalisation in natural language	
Event	**System receives the scores from each module of Co-Inform technologies. Each module gives a score:** Example: *similarity(post) -> score_1* *semantic(post) -> score_2* *stance(post) -> score_3* *veracity(post) -> score_4* *credibility(post) -> score_5* *...*
Conditions	**Each module has different threshold. One or more scores are above threshold:** Example: filter ->* *score_1 > threshold_1* *score_2 > threshold_2* *score_3 > threshold_3* *score_4 > threshold_4* *score_5 > threshold_5* *...* *filter(all scores > their thresholds) -> filtered_scores*
Actions	**Show critical scores above threshold:** *show(filtered_scores)*

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix
Flagging Policy #1: Showing the most critical score above threshold**.**

Risk:
The subjectiveness of selecting an initial and revised threshold level might present a risk.

ii. Flagging Policy #2: See more, see less button

This policy is one of the flagging policies. The system shows the top *n* scores where *n* is the number of scores. The user can see more/less results by clicking the "see more, see less" button. User sets top *n* scores from the settings of the tool.
An example use case of this policy would be to hide the veracity of a claim and/or user confirmation biases, unless the user clicks "see more" button.

ECA formalisation in natural language:

Table 2. Flagging Policy #2 ECA formalisation in natural language	
Event	*filter(all scores > their thresholds) -> filtered_scores* *count(filtered scores) -> num_critical_scores*
Conditions	*num_critical_scores > n && clickedSeeMore == true*
Actions	*getPrioritisedScores(filtered_scores) -> prioritised_scores* *expand(prioritised_scores)*

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix **Flagging Policy #2: See more/See less,**

Risk:
The subjectiveness of the user choosing "more/less" might present a risk.

iii. Flagging Policy #3: Notify user

As
Flagging Policy #1: Showing the most critical score above threshold, the system detects a misinforming post based on responses from the modules, and then the system notifies the user by showing the number of new notifications. User can change when to be notified about a misinforming post by setting the thresholds. A notification occurs when the user settings are met. When the user clicks on the notification button, it will show a menu with the relevant notification for the post.

ECA formalisation in natural language:

Table 3. Flagging Policy #3 ECA formalisation in natural language	
Event	*filter(all scores > their thresholds) -> filtered_scores* *count(filtered scores) -> num_critical_scores*
Conditions	*num_critical_scores > n && enableNotification == true*
Actions	*notify(user)*

ECA formalisation in rule engine:

An example marked with rule language is given in Appendix **Flagging Policy #3: Notify User.**

Risk:
Lots of alerts and notifications can be annoying for users, and there is a risk that users turn off the notifications.

4.2. Marking Policy

As Flagging Policy Flagging Policy, the system receives scores from each module. When the scores indicate a possibility for misinforming content, the plugin marks the post. Possible policies for marking the posts are as follows:

i. Blurring the whole post

When the content has misinforming scores from at least *n* modules where *n* indicates the threshold, the system blurs the whole post.

ECA formalisation in natural language:

Table 4. Marking Policy #1 ECA formalisation in natural language	
Event	*count(filtered scores) -> num_critical_scores*
Conditions	*If num_critical_scores >= n*
Actions	*blur(post, blurring_percentage = 100)*

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix Marking Policy #1 and #2: Blurring post full/partially.
Risk:
Blurring the whole post could contribute to the interestingness of the post, and hence it promotes the misinformation. To avoid this risk, the post might be blurred only when the cursor is on top of the post (see Marking Policy #1 and #2: Blurring post full/partially.

ii. Blurring the whole post when the cursor is on top of the post

When the content has misinforming scores from at least *n* modules where *n* indicates the threshold and the cursor is on top of the post, the system blurs the whole post.

ECA formalisation in natural language:

Table 5. Marking Policy #2 ECA formalisation in natural language	
Event	*count(filtered scores) -> num_critical_scores*

Conditions	*If num_critical_scores >= n && cursorOnPost*
Actions	*blur(post, blurring_percentage = 100)*

ECA formalisation in rule engine:
An example marked with rule language is similar to the one given in Appendix Marking Policy #1 and #2: Blurring post full/partially. The only difference is that an additional condition is required.

iii. Blurring the post partially

When the content has misinforming scores from at least *n* modules where *n* indicates the threshold, the policy changes the resolution of the content such that it remains readable, but its quality is affected.

ECA formalisation in natural language:

Table 6. Marking Policy #3 ECA formalisation in natural language	
Event	*count(filtered scores) -> num_critical_scores*
Conditions	*If num_critical_scores >= n*
Actions	*blur(post, blurring_percentage = x)*

ECA formalisation in rule engine:
ECA formalisation is similar to policy Marking Policy #1 and #2: Blurring post full/partially.

iv. Blurring the retweet button

When the content has misinforming scores from at least *n* modules where *n* indicates the number of critical scores, the system blurs the retweet button.

ECA formalisation in natural language:

Table 7. Marking Policy #4 ECA formalisation in natural language	
Event	*count(filtered scores) -> num_critical_scores*
Conditions	*If num_critical_scores >= n*
Actions	*blur(post.retweet_button)*

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix Marking Policy #3: Blurring retweet button.

Risk:

This policy only informs the user when the user tries to retweet. There is a risk that the user might not be informed about the reliability of the post.

v. Putting a visual stop sign on top

When the content has misinforming scores from at least *n* modules where *n* indicates the threshold, the system places a stop sign on top in order to warn the user.

ECA formalisation in natural language:

Table 8. Marking Policy #5 ECA formalisation in natural language	
Event	*count(filtered scores) -> num_critical_scores*
Conditions	*If num_critical_scores >= n*
Actions	*place(post, stop_sign)*

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix Marking Policy #4: Putting a visual stop sign.

Risk:
Similar to Marking Policy #1 and #2: Blurring post full/partially, a stop sign could contribute to the interestingness of the post, and promote the misinformation.

vi. Putting circle information icon on the top

When the content has misinforming scores from at least *n* modules where *n* indicates the threshold, the system places a circle information icon with the words “Potential Misinformation” and a button that leads to an explanation on why the post is misinforming.

ECA formalisation in natural language:

Table 9. Marking Policy #6 ECA formalisation in natural language	
Event	*count(filtered scores) -> num_critical_scores*
Conditions	*If num_critical_scores >= n*
Actions	*place(post, circle_information_icon, explanation_button)*

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix Marking Policy #5: Putting a circle information icon on the top.

Risk:
N/A

vii. Removing a post

When the content has misinforming scores from at least *n* modules where *n* indicates the threshold, the system removes the post.

ECA formalisation in natural language:

Table 10. Marking Policy #7 ECA formalisation in natural language	
Event	*count(filtered scores) -> num_critical_scores*
Conditions	*If num_critical_scores >= n*
Actions	*remove(post)*

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix Marking Policy #6: Removing a post.

Risk:
System might not have a grant to remove the post from the social media. As a solution, the post might be reported to the corresponding social media by the system.

4.3. Reporting Policy

The goal of reporting policy is to improve the functional modules of the platform by getting feedback from users and to detect spammer users of the platform by analysing those feedbacks.

i. Reporting not flagged post

User can report a post that should be flagged by the system. When the user reports a post, the system sends a form. When the form is filled by the user, it is saved for analysis by admins of the system.

ECA formalisation in natural language:

Table 11. Reporting Policy#1 ECA formalisation in natural language	
Event	**User reports a post:** *user.report(post)*
Conditions	*post.status = not flagged*
Actions	**System sends a report:** *system.sendForm(user) -> form* *user.fill(form) -> filled_form* **When form is filled by the user, the form is saved for analysis by admins of the system:**

	system.save(filled_form, user)

According to the condition, the user receives the following form to report the post that is not flagged by the system:

The form which will be shown to the user:

Table 12. Form for flagging the post
Why would you flag this content, because it is: ● *Satire/Parody* ● *Misleading content* ● *Imposter Content* ● *Fabricated Content* ● *False Connection* ● *False Context* ● *Conspiracy Theory/Rumor* ● *Private information* ● *Other* ● Can you provide corrective evidence? (optional) Attach ---------- (url)

Definitions of options given in the form are as follows:

- **Satire or Parody:** No intention to cause harm but has potential to fool
- **Misleading Content**: Misleading use of information to frame an issue or individual
- **Imposter Content:** When genuine sources are impersonated
- **Fabricated Content:** News content is 100% false, designed to deceive and do harm
- **False Connection:** When headlines, visuals or captions don't support the content
- **False Context:** When genuine content is shared with false contextual information
- **Manipulated Content:** When genuine information or imagery is manipulated to deceive
- **Private Information:** Leak of privacy information
- **Other:** The user can select this option if none of the above options is the reason of report. When the user selects the other option, the system takes a textual input from the user.

When the user fills the form for flagging a post, for each of the options, a pop-up definition will be shown to the user.

ECA formalisation in rule engine:

An example marked with rule language is given in Appendix Reporting Policy #1: Reporting not flagged post.

Risk:
User might be a spammer (e.g. bot) who reports the posts several times. In this case, feedback analysis by admins of the system will enable to detect these users. Additionally, user policies described in Spammer user's policy will protect the system from spammers.

ii. Reporting flagged post

User can report a post that might have been mistakenly flagged by the system and bot/spammer. Similar toReporting Policy #1: Reporting not flagged post, when the user reports a post, the system sends a form. When the form is filled by the user, the form is saved for analysis by admins of the system.

ECA formalisation in natural language:

Table 13. Reporting Policy #2 ECA formalisation in natural language	
Event	User report a post *user.report(post)*
Conditions	*post.status = flagged*
Actions	System sends a report *system.sendForm(user) -> form* *user.fill(form) -> filled_form* When form is filled by the user, the form is saved for analysis by admins of the system. *system.save(filled_form, user)*

Form for posts that might be mistakenly flagged by the system

Table 14. Form for unflagging posts
Why would you unflag this content, because it is: ● Unverifiable Claim ● Opinion ● Known Satire/Parody ● Fact-checked ● New Evidence ● Other Can you provide evidence? (optional) [URL]

Definitions of options given in the form are as follows:

- **Unverifiable Claim:** This post contains a claim that cannot be verified or debunked.
- **Opinion:** This post represents an expression of personal or political opinion or belief.

- **Known Satire/Parody:** This post contains material from a self-declared satire/parody publisher [Check against a whitelist]
- **Fact-checked:** This post has been reviewed by professional fact checkers." [Check against database of fact-check URLs.]
- **New evidence:** New evidence is available to review this post's claim(s).
- **Other:** The user can select this option if none of the above options is the reason of report. When the user selects the *other* option, the system takes a textual input from the user.

ECA formalisation in rule engine:
An example marked with rules language is given in Appendix Reporting Policy #2: Reporting flagged post.

Risk:
The same risk as for Reporting Policy #1: Reporting not flagged post applies for this policy.

4.4. User Management Policy

User management policy protects the system from spammer users.

i. Spammer user's policy

If a user requests to flag or not flag more than *x* times in the past *d* days, the system checks if the user provided evidence for posts. If not, the user will be reported as a spammer, and the system will not accept requests from users that seem to be spammers.

ECA formalisation in natural language:

Table 15. User Management Policy #1 ECA formalisation in natural language	
Event	num_flagging_post > x
Conditions	If num_evidence_post < n
Actions	system.changeStatus(user) -> spammer system.changeFlagging(user) -> disable

ECA formalisation in rule engine:
An example marked with rule language is given in Appendix User Management Policy.

Risk:
The System mistakenly classifies a decent user as a spammer.

5. Deployment of Policies

As we described in D2.1, policies will be registered in a rule engine. The rule engine described in D2.1 was a semantic rule engine. In order to cope with the real-time stream of misinformation, we have updated the rule engine with an open source ECA rule engine[7]. The updated rule engine supports the execution of JSON-formatted rules. We provided machine readable versions of each policy, which are compatible with the updated rule engine, in the Appendix.

The rule engine will run as a module of the Policy framework[8]. The policy framework will be integrated into the Co-Inform platform. The policy framework will communicate with each module for deciding actions under conditions.

[7] https://github.com/j-easy/easy-rules

[8] https://github.com/isspek/policy_manager

6. Conclusion

In this deliverable, we have described Co-Inform platform policies with both human-readable and machine-readable formalisations and provided their risks. Additionally, we explained how the policies will be evaluated in the upcoming co-creation workshops.

Co-Inform platform policies aim for the orchestration of tools developed within WP3 and WP4 in an automated manner. As the next step, machine readable formulas of each policy will be registered to the open-sourced ECA rule engine that is part of the policy framework. Finally, the Co-Inform platform prototype, along with the policies, will be tested and evaluated at the workshops.

7. References

Curtis, A. M., Wells, T. M., Lowry, P. B., & Higbee, T. (2008). An Overview and Tutorial of the Repertory Grid Technique in Information Systems Research. *Communications of the Association for Information Systems*, *23*(November). https://doi.org/10.17705/1cais.02303

Bae, J., Bae, H., Kang, S. H., & Kim, Y. (2004). Automatic control of workflow processes using ECA rules. *IEEE transactions on knowledge and data engineering*, *16*(8), 1010-1023.

Appendix

Flagging Policy

i. Flagging Policy #1: Showing the most critical score above threshold


```
{
 "name": "flagging_similarity_rule",
 "description": "similarity above threshold",
 "condition": "similarity >= threshold_similarity",
 "actions": [
   "callback.showAnalysis(\"similarity\", similarity);",
   "critical_score_count = critical_score_count + 1;",
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);"
 ]
},
{
 "name": "flagging_semantic_rule",
 "description": "semantic above threshold",
 "condition": "semantic >= threshold_semantic",
 "actions": [
   "callback.showAnalysis(\"semantic\",semantic);",
   "critical_score_count = critical_score_count + 1;",
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);"
 ]
},
{
 "name": "flagging_stance_rule",
 "description": "stance above threshold",
 "condition": "stance >= threshold_stance",
 "actions": [
   "callback.showAnalysis(\"stance\",stance);",
   "critical_score_count = critical_score_count + 1;",
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);"
 ]
},
{
 "name": "flagging_veracity_rule",
 "description": "veracity above threshold",
 "condition": "veracity >= threshold_veracity",
 "actions": [
   "callback.showAnalysis(\"veracity\",veracity);",
   "critical_score_count = critical_score_count + 1;",
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);"
 ]
},
{
 "name": "flagging_credibility_rule",
 "description": "credibility above threshold",
 "condition": "credibility >= threshold_credibility",
```

```
 "actions": [
   "callback.showAnalysis(\"credibility\",credibility);",
   "critical_score_count = critical_score_count + 1;",
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);"
 ]
}
```

ii. Flagging Policy #2: See more/See less

```
{
 "name": "flagging_policy_see_more_see_less",
 "description": "flagging policy: see more see less",
 "condition": "critical_score_count >= threshold_num_critical_scores &&
button.clickedSeeMore== True",
 "actions": [
   "callback.expand(postId)"
 ]
}
```

iii. Flagging Policy #3: Notify User

```
{
 "name": "flagging_policy_notify_user",
 "description": "flagging policy: notify user",
 "condition": "critical_score_count >= threshold_num_critical_scores &&
system.enabledNotification == True",
 "actions": [
   "callback.notify(userId)"
 ]
}
```

Marking Policy

i. Marking Policy #1 and #2: Blurring post full/partially

```
{
 "name": "marking_policy_blurring_post",
 "description": "marking policy: blurring the post",
 "condition": "critical_score_count >= threshold_num_critical_scores",
 "actions": [
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);",
   "callback.blur(postId, threshold_blur_percentage);"
 ]
}
```

ii. Marking Policy #3: Blurring retweet button

```
{
 "name": "marking_policy_blurring_retweet_button",
 "description": "marking policy: blurring the retweet button",
 "condition": "critical_score_count >= threshold_num_critical_scores",
 "actions": [
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);",
   "callback.place(postId,\"retweet_button\" );"
 ]
```

```
}
```

iii. Marking Policy #4: Putting a visual stop sign

```
{
 "name": "marking_policy_visual_stop_sign",
 "description": "marking policy: putting a visual stop sign",
 "condition": "critical_score_count >= threshold_num_critical_scores",
 "actions": [
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);",
   "callback.place(postId, stop_sign_icon);"
 ]
}
```

iv. Marking Policy #5: Putting a circle information icon on the top

```
{
 "name": "marking_policy_putting_circle_information",
 "description": "marking policy: putting a circle information icon on the top",
 "condition": "critical_score_count >= threshold_num_critical_scores",
 "actions": [
   "callback.place(postId, location = top, information_icon, explanation_button);"
 ]
```

v. Marking Policy #6: Removing a post

```
{
 "name":"marking_policy_putting_a_visual_stop_on_top",
 "description":"marking policy: placing a stop sign in order to warn the user",
 "condition":"critical_score_count >= threshold_num_critical_scores",
 "actions":[
   "callback.showAnalysis(\"critical_score_count\", critical_score_count);",
   "callback.place(postId,\"stop_sign\" );"
 ]
}
```

Reporting Policy

i. Reporting Policy #1: Reporting not flagged post

```
{
 "name": "reporting policy",
 "description": "reporting policy: reporting not flagged post",
 "condition": "post.flagged == False",
 "actions": [
   "callback.startWorkflowReportNotFlagging(userId,postId);"
 ]
}
```

ii. Reporting Policy #2: Reporting flagged post

```
{
 "name": "reporting policy",
 "description": "reporting policy: reporting flagged post",
 "condition": "post.flagged == True",
```

```
 "actions": [
   "callback.startWorkflowReportNotFlagging(userId,postId);"
 ]
}
```

User Management Policy

i. Spammer users policy

```
{
 "name": "user_management_spammer_detection",
 "description": "user management policy: user management policy protects the system from spammer users. ",
 "condition": "userGroup.equals(\"normal\") && num_flag >= threshold_num_flagging_post && num_evidence <= threshold_num_evidence_post",
 "actions": [
   "userGroup = \"spammer\";"
 ]
},
{
 "name": "user_management_spammer_permission",
 "description": "user management policy: user management policy protects the system from spammer users. ",
 "condition": "userGroup.equals(\"spammer\")",
 "actions": [
   "callback.restrict(userId, \"tagging\");"
 ]
}
```