Sensibilité aux situations de façon collaborative

Michal Szczerbak

N° d'ordre : 2013telb0298

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En accréditation conjointe avec l'Ecole Doctorale Matisse

Ecole Doctorale – MATISSE

COLLABORATIVE SITUATION AWARENESS

Thèse de Doctorat

Mention : Informatique

Présentée par **Michał Krzysztof Szczerbak**

Département : RSM

Directeur de thèse : Jean-Marie Bonnin

Soutenue le 18 octobre 2013

Jury :

Anne-Marie Kermarrec	Dir. de rech.	INRIA	Rennes	(Présidente)
Noël Crespi	Professeur	Telecom Sud Paris	Paris	(Rapporteur)
Isabelle Mirbel	M. de conf.	Université Nice Sophia Antipolis		(Rapporteuse)
Jean-Marie Bonnin	Professeur	Telecom Bretagne	Rennes	(Directeur de thèse)
Ahmed Bouabdallah	M. de conf.	Telecom Bretagne	Rennes	(Encadrant de thèse)
François Toutain	Docteur	Orange Labs	Lannion	(Encadrant de thèse)
Fiora Pirri	Professeur	Università di Roma	Rome	(Examinatrice)
Azim Roussanaly	M. de conf.	LORIA	Nancy	(Examinateur)
Jean-Michel Portugal		Orange Labs	Lannion	(Invité)

Problématique

On demande souvent à la technologie d'être un prolongement naturel du comportement des utilisateurs. L'adaptation des différentes solutions techniques devrait permettre idéalement de simplifier les activités humaines dans leurs formes originales.

L'objectif de notre recherche consiste à étudier comment un système pourrait augmenter la compréhension par un utilisateur de sa situation au sens large, pour lui permettre d'améliorer de façon pertinente sa prise de décision.

Ainsi une décision à prendre dans une situation particulière pourrait être technologiquement assistée. L'intérêt d'un tel soutien technologique apparaît clairement dans le cas des situations que l'utilisateur n'a pas encore rencontrées et a propos desquelles il ne dispose encore d'aucune expérience.

Le comportement naturel humain consiste à se renseigner auprès d'autres personnes ayant une expérience de la situation en question. Ce type d'échange constitue l'essence même de l'intelligence collective de la communauté.

D'après les chercheurs, la coopération et la compétition dans les groupes permettent l'émergence d'une nouvelle intelligence collective hissant l'ensemble du groupe au degré de compréhension de ses membres les plus avisés.

Le même principe pourrait être appliqué à un système artificiel. Il existe dans ce sens des systèmes de recommandation basés sur les techniques de filtrage collaboratif, mais aucun d'eux ne traite du domaine de la sensibilité au contexte et plus généralement à la situation.

Nous introduisons le système KRAMER comme un outil de soutien aux décisions de ses utilisateurs basé sur l'appréciation de la situation courante. Il détermine les suggestions d'actions et l'importance des situations en s'appuyant sur l'ensemble de ses utilisateurs.

Le principe de notre approche

Le filtrage collaboratif, qui constitue une des solutions techniques implémentant l'intelligence collective, est déjà utilisé avec succès dans plusieurs systèmes de recommandation, par exemple dans les boutiques web qui suggèrent les articles à acheter en fonction de nos goûts et ceux des personnes qui nous ressemblent.

Actuellement la prise en compte de la situation de l'utilisateur (son contexte) par les systèmes, n'est considérée que pour améliorer et raffiner la liste des objets recommandés. Ni le contexte, ni la situation ne sont eux-mêmes considérés comme les objets possibles d'une telle recommandation.

C'est néanmoins l'hypothèse de base de notre recherche : une situation contextuelle peut constituer l'objet de la recommandation. Les situations suggérées dans ce cas

apporteraient de plus une nouvelle connaissance d'ordre supérieure sur la communauté des utilisateurs, en identifiant les situations importantes pour elle. Cette connaissance pourrait de plus être exploitée par différents services à la communauté.

La communication interpersonnelle constitue le point de départ de nos travaux. Comme nous l'avons déjà mentionné, le processus d'échange des conseils constitue la structure de communication de base pour l'intelligence collective dans les communautés humaines. Ainsi notre système en suggérant des situations importantes à la communauté des utilisateurs, devrait leurs permettre de réagir de façon éclairée à ces situations.

Nous partons du phénomène du microblogging, que nous sommes une société toujours connectée et que nous partageons de plus en plus d'informations. Ces informations sont souvent très personnelles et leur ensemble est très informatif, mais elles sont très peu structurées. Ainsi la connaissance stockée dans le Web existe déjà, il faudrait cependant pouvoir la convertir en connaissance collective.

Le fait de traiter des situations au lieu des objets physiques a des conséquences importantes. Tout d'abord, une situation nécessite un modèle. Nous considérons de plus la multi-dimensionnalité du contexte, ainsi que le fait qu'une situation est souvent influencée par le contexte de nos proches. Nous cherchons finalement un modèle expressif et sémantiquement composé.

Cela implique aussi, que le traitement des situations nécessite des mécanismes de raisonnement sémantique pour renforcer la nature statistique du filtrage collaboratif classique. En conclusion nous introduisons un nouveau modèle de situation et un algorithme innovant pour traiter ces situations afin de proposer un nouveau médium d'informations social.

L'implémentation

Le premier pas pour implémenter le système est sa décomposition. Suggérer des situations importantes nécessite trois éléments principaux : le fournisseur des suggestions, leur destinataire, et le modèle de situations, qui seront expliqués plus tard dans notre section. Le serveur implémentant le premier point, est le cœur de notre solution et il utilise notre modèle de situations.

Le service est destiné aux utilisateurs qui contribuent avec leurs propres informations à notre système, devenant dans le même mouvement les destinateurs de l'intelligence dérivée en retour. Du fait qu'il s'agit de données et de suggestions très personnelles et socialement significatives, cette partie du système est implémentée sous la forme d'un logiciel sur téléphone mobile.

Les smartphones aujourd'hui ne sont pas seulement des outils pour réaliser des appels. Ils sont aussi capables d'échanger des données de plusieurs types, de

capturer les données de contexte, et d'analyser et de traiter l'ensemble de ces données. Par ailleurs, ils ont une nature très personnelle. Dans le cas du domaine de communication interpersonnelle nous employons l'ensemble de ces capacités comme une extension d'un carnet d'adresse sensible au contexte.

Du point vue de KRAMER, ce module client de notre système a comme objectif (1) alimenter KRAMER avec des règles de déclenchement d'une action suite à une situation, (2) recevoir des suggestions des règles venant de KRAMER, et (3) enregistrer un feedback d'utilisateur afin de réévaluer ces règles. Par ailleurs, d'un point vue d'utilisateur, (4) l'interface du logiciel doit permettre de définir de nouvelles règles et (5) l'ensemble des situations définies et reçues doit déclencher les actions des règles.

Il est aussi requis l'existence d'un système de partage de contexte entre proches, qui permette de voir ce contexte dans le carnet d'adresse, d'ajouter ce contexte dans les situations définies, et de déclencher les actions en fonction de ce contexte et des règles. Nous présentons une implémentation possible d'un tel système, ainsi que les autres blocs fonctionnels du module, dans le mémoire. Ceci détermine par ailleurs le périmètre de notre recherche. Nous limitons notre prototype aux notifications ciblant des actions dans les règles.

Définir une situation importante, en recevoir une nouvelle comme une suggestion, et déclencher à partir de ces dernières une notification en retour, nécessite déjà d'avoir un modèle de situation. Une situation constitue pour nous un objet composé de plusieurs dimensions de contexte (localisation, disponibilité, etc.) pour plusieurs contacts du carnet d'adresse (plusieurs entités dans un cas plus général que la communication interpersonnelle).

Nous créons ce modèle en employant les arbres conceptuels, dans lesquels les nœuds portent les concepts sémantiques pour chaque dimension de contexte et les arcs définissent les types de ces dimensions. On obtient une composition de plusieurs triplets de contexte (dimension, relation, valeur) en une structure plus expressive. Chaque concept dans l'arbre provient d'une taxonomie correspondante à la dimension, et peut modéliser une partie de la réalité sur un quelconque niveau d'abstraction.

L'interface du module client permet de définir des situations aussi complexes. Nous avons implémenté un mécanisme qui peut comparer la situation courante avec l'ensemble des situations définies dans les règles, afin de déclencher la notification associée. Pour cela nous adaptons le système de production Rete pour qu'il soit capable de reconnaître une valeur de contexte sur différents niveaux d'abstraction.

D'un point de vue abstrait, ce mécanisme doit déterminer si deux situations sont en relation où si l'une est plus abstraite que l'autre, qui est en revanche une version plus détaillée de la première. Notre modèle de situation permet de le faire simplement,

l'ensemble des situations constituant un treillis de situations organisé par la relation plus abstrait / plus détaillé.

Le treillis est lui-même un objet abstrait, construit à partir des arbres conceptuels et des règles de formation des graphes conceptuels de Sowa. Par exemple, la restriction sémantique d'une valeur dans un nœud d'un graphe conduit à une situation plus détaillée. De même pour la fusion de deux graphes. Le treillis obtenu ainsi permet de réduire l'opération de détection d'une situation à celle de l'exploration de ce treillis.

Le cœur de notre système, le serveur KRAMER, utilise le même modèle de situation, basé sur les mêmes taxonomies que ses clients. Il reçoit l'ensemble des règles de notification, et a une connaissance de l'ensemble des situations associées. Son objectif est de les traiter et d'en suggérer une partie aux utilisateurs. Le choix des situations à recommander à un utilisateur particulier est une fonction de l'importance de ces situations et de la situation courante.

C'est ici que le système dérive l'intelligence collective, que nous implémentons par le filtrage collaboratif. Nous trouvons par contre que le traitement sémantique est beaucoup plus pertinent dans notre cas, que le traitement statistique, qui est employé d'habitude dans les boutiques web, par exemple.

Cela constitue l'une des hypothèses que notre recherche voulait vérifier : l'importance d'une situation à l'échelle d'une communauté serait mieux déterminée si le système KRAMER parvient à identifier les groupes de situations sémantiquement similaires.

Par exemple, en fonction de l'utilisateur on peut définir les deux situations suivantes : « le fils est à l'école » et « la fille est à l'école », qui sont toutes deux des versions concrètes de la situation plus abstraite « l'enfant est à l'école ». Les coupler permettrait d'une part d'éviter de traiter ces situations comme différentes et d'autre part d'augmenter leur nombre d'occurrences en les rendant plus populaires. Le même principe s'applique à toutes les dimensions de contexte.

Nous introduisons un algorithme de traitement des situations exprimées par notre modèle des arbres conceptuels, qui généralise l'ensemble des situations pour en trouver celles qui sont les plus importantes pour la communauté d'utilisateurs. Cette généralisation est par nature opportuniste, dans le sens où les niveaux d'abstraction obtenus dépendent des situations à l'entrée.

L'algorithme est composé de deux étapes. D'abord l'ensemble des situations est divisé en groupes des situations ayant la même structure vis-à-vis du modèle de situations. Ceci est une opération qui utilise une formule récurrente présentée ci-dessous, qui compare les structures de deux situations. La formule retourne « 1 », si les structures sont les mêmes.

La deuxième étape prend chaque groupe obtenu séparément et y applique la généralisation qui est l'innovation principale de cette partie du système KRAMER.

Nous utilisons le prunning des arbres sur une structure du type méta-arbre, représentant la composition de l'ensemble des situations traitées à la fois. En conséquence toutes les situations dont tous les concepts sont sémantiquement proches et n'ayant pas disparu après le processus du prunning, sont généralisées en une situation.

L'algorithme décrit s'adapte également aux retours d'utilisateurs ayant reçu les situations suggérées. Une fois la notification déclenchée suite à une situation recommandée, l'utilisateur peut décider de garder la règle ou la rejeter. Quelle que soit la décision, elle est prise en compte par le système, influençant ainsi le processus de généralisation et permettant l'adaptation du système aux besoins d'utilisateurs, qui eux changent en fonction du temps. L'ensemble du processus peut aussi être interprété comme une manipulation du treillis des situations.

Expérimentations

Afin d'évaluer le système KRAMER et ses différents paramètres, nous avons réalisé deux tests. Le premier étant une simulation du comportement du système et son passage à l'échelle (scalabilité). Le deuxième étant un test d'utilisateurs de petite dimension qui a permis d'avoir un retour d'expérience sur le système.

Grâce au premier test nous avons appris surtout que la fonction de scalabilité de l'algorithme de généralisation peut être considérée linéaire. Nous avons exploré également l'impact de plusieurs facteurs sur cette fonction, par exemple le nombre des dimensions de contexte, le ratio de feedback négatif dans le système, etc. Mais pour des configurations de taille raisonnable le traitement des situations ne prend guère plus que quelques secondes.

Un des résultats de cette partie des tests nous a permis d'observer, est que le système n'est pas robuste quant à la quantité de situations aléatoires. Notre algorithme, qui ne permet pas d'obtenir une situation généralisée avec des concepts les plus généraux possibles, a la spécificité suivante : il produit dans les conditions extrêmes, un nombre de situations généralisées supérieur au nombre de situations en entrée.

Nous avons réalisé un second test ciblant l'utilisation réelle du système. Afin d'avoir les résultats initiaux très vite, deux groupes de testeurs, 8 personnes dans chacun d'entre eux, ont participé à un test scénarisé. En effet nous avons construit un jeu du type chasse au trésor, où chaque objectif accompli était associé avec une règle introduite dans le système.

Ce jeu a permis à la plupart des participants d'accomplir un ensemble d'objectifs, en un nombre de tours largement inferieur à ce que nous avons évalué en l'absence de soutien par un système comme KRAMER. Une décision sur trois dans tout ce jeu a été un effet des suggestions du système. Nous avons acquis cette information, ainsi

que l'acceptation générale de l'idée, depuis les logs du système et des questionnaires hors-ligne.

Conclusions

Le système KRAMER que nous avons proposé comme résultat de notre recherche, est un outil de support de décisions éclairées, sensible au contexte et aux situations. C'est aussi un outil social d'interaction interpersonnelle, qui augmente l'intelligence collaborative de ses utilisateurs qui contribuent collectivement à déterminer les situations importantes, en les fournissant et les réévaluant.

Même si notre prototype a été adapté au scénario de communication, KRAMER peut être facilement adapté à un domaine quelconque dans lequel le problème est défini par des graphes conceptuels et dans lequel la logique de déclenchement d'actions exploiterait une solution bénéficiant d'une approche collective. La domotique en constitue un exemple intéressant.

D'une point vue implémentation du système, nous apportons deux blocs innovants. Le premier étant le modèle de situations comme des arbres conceptuels. Cela nous conduit plus loin à définir un treillis de situations, qui simplifie les manipulations sur l'abstraction des situations. Le deuxième étant l'algorithme de généralisation sémantique des arbres conceptuels.

Nous avons effectué deux types de tests, dont l'un relativement innovant. Cela nous a permis d'évaluer le système de façon globalement positive. Il est clair par contre qu'un test d'utilisateurs grandeur nature sera nécessaire pour améliorer KRAMER à des fins notamment de commercialisation. Ce point va constituer une suite ultérieure au travail de cette thèse.

Par ailleurs, le filtrage d'informations dans KRAMER n'est basé que sur les mécanismes sémantiques, alors que les applications classiques sont plutôt du type statistique. Cet aspect des similarités des utilisateurs du système reste encore à incorporer dans KRAMER et les conclusions sur une éventuelle amélioration du système de cette manière ne seront pas connues avant le test mentionné précédemment.

De plus, le fait d'adapter KRAMER à d'autres domaines nécessite d'analyser l'impact des actions associées aux situations dans les règles de ce système. Notamment, dans le domaine de la domotique nous imaginons avoir des règles contradictoires. Cela doit être traité pour que chaque implémentation de KRAMER soit fonctionnelle.

Finalement, il existe également d'autres travaux de recherche à effectuer pour différentes extensions du système. Ceci ne contredit pas que KRAMER et notre recherche en général apportent une valeur scientifique significative pour les domaines de recherche suivants : l'intelligence collective, la sensibilité aux situations, et le « social computing ».

"Someday my log will have something to say about this."

The Log Lady

Elizie, za każdy wspólny dzień

Table of contents

About the author - 4 -
Acknowledgements - 5 -
List of figures and tables - 6 -
Glossary - 8 -
Abstract - 10 -
1. Problem statement - 13 -
2. Theoretical background - 16 -
2.1. Situation cognition - 16 -
2.1.1. Situation Theory - 17 -
2.1.2. Situation models - 18 -
2.1.3. Situation awareness - 21 -
2.2. Decision support - 22 -
2.2.1. Recommender systems - 22 -
2.2.2. Contextual systems - 24 -
2.2.3. Cooperative systems - 25 -
3. Proposed solution - 27 -
3.1. Working hypothesis - 27 -
3.2. Informed decisions - 28 -
3.3. Functional decomposition - 30 -
4. System implementation - 32 -
4.1. Chosen domain - 32 -
4.1.1. Interpersonal relations - 32 -
4.1.2. Target group - 34 -
4.1.3. Use cases - 34 -
4.2. Situations model - 37 -
4.3. System architecture - 42 -
4.4. COSMO client - 45 -
4.4.1. Context sharing - 46 -
4.4.2. Situations exchange - 51 -
4.4.3. Action execution - 54 -
4.5. Communication protocol - 59 -
4.6. KRAMER server - 62 -
4.7. Important situations - 63 -
4.7.1. Situation popularity - 64 -
4.7.2. Generalization algorithm - 67 -
4.7.3. Rules revaluation - 74 -
5. Evaluation tests - 78 -
5.1. Algorithms simulations - 80 -
5.2. Treasure-hunt game - 91 -
5.3. Result discussion - 99 -
6. System applications - 103 -
6.1. Social computing - 104 -
6.2. Home automation - 106 -
7. Final thoughts - 108 -
List of publications - 111 -
List of patents - 112 -
References - 113 -

About the author

I was born and raised in Warsaw, Poland in 1985. After obtaining my MSc degree at the Electronics and Information Techniques department of Warsaw School of Technology (Politechnika Warszawska) in 2010, my passion for telecommunication technologies and intelligent services led me to Orange Labs, the research and development branch of France Telecom, where I started my PhD research. I have to admit that the conceptual work has been inspired greatly by those calm and savage coasts of a beautiful region of Breizh, to where I have moved in consequence.

Other from the office hours, I love to spend my spare time with my family, lately especially with a little baby boy, who made the last several months of the research so much more interesting. Moreover, in the scenery of Armorica countryside I try to teach my dog to behave, I watch over and over my favourite movies, I complain that I have already read all the good books there are, and I play an insane amount of board games, hoping that the latter would become an Olympic discipline.

In the future I would like to devote myself more to breeding dogs and goats, transforming the forces of nature into heat, and ultimately to doing everything in power for my household to become a self-sufficient farm. While still contributing to the world of science, naturally.

Acknowledgements

The research work of this thesis were carried out in the scope of the CIFRE (Convention industrielle de formation par la recherche) contract in cooperation between Telecom Bretagne university and Orange Labs research department of Orange telecom operator under subvention of the French Government (grant number 1067/2010).

I would like to thank my supervisors from both Orange Labs and Telecom Bretagne, namely Dr. François Toutain and Dr. Ahmed Bouabdallah. I appreciate the liberty that you gave me, which motivated my independent getting into the bottom of the problem. Thanks to you I could follow my own hunches for finding a solution to the research problem you had sketched at the beginning. I am equally thankful to the thesis director, Prof. Jean-Marie Bonnin, who has assured good framing and structure of my work for the past three years.

Moreover, I would like to express my gratitude to the Orange Labs team I was included in, with Christophe Dejouhanet in charge of it, as well as my other fellow workers from the hallway. Their warm welcome and everyday kindness made the experience even more enjoyable. I shall also remember interacting with the research projects leaders, namely Jean-Michel Portugal, Emmanuel Le Huerou, and Claude Daloz, and other experts, whose interest in the results of my research made it feel useful in the end. Special thanks to Dominique Deuff for preparing a user test of the KRAMER system.

I acknowledge also everyone who helped in the process of driving this research from the very beginning to the very end; everyone expressing positive criticism, especially the dissertation reviewers, and arranging the day of the thesis defence, especially the assistants at the RSM department. I am also grateful to all jury members for accepting the invitation.

Finally, who does one thank for creating that gorgeous Armorica?

List of figures and tables

Figure 2-1. Endsley's model of situation awareness - 22 -
Figure 2-2. Paradigms for incorporating context in recommender systems - 24 -
Figure 2-3. User influenced by others directly or through derived intelligence - 25 -
Figure 2-4. The collaborative filtering process - 26 -
Figure 3-1. The KRAMER system transferring an experience of a group to a user - 28 -
Figure 3-2. A situation of a user is a composition of his relatives and their context - 29 -
Figure 3-3. Functional decomposition of the KRAMER system - 30 -
Figure 3-4. Client-server architecture inferred from the functional blocks - 31 -
Figure 4-1. Phonebook 2.0 Android application screen - 33 -
Figure 4-2. The meta-model of situations - 37 -
Figure 4-3. Inferring a conceptual tree from the situation meta-model - 38 -
Figure 4-4. Every concept in a modeled situation comes from the respective … - 39 -
Figure 4-5. A modeled situation is a partial representation of the real one - 39 -
Figure 4-6. Situation specialization with two possible operators shown on … - 41 -
Figure 4-7. A sample part of a situation lattice - 42 -
Figure 4-8. The architecture of our prototype, including the context … - 43 -
Figure 4-9. Social relations in the context distribution system - 44 -
Figure 4-10. Two levels of cooperation in KRAMER, links are formed between … - 44 -
Figure 4-11. The generic COSMO architecture - 45 -
Figure 4-12. Structure of the context distribution system and its relation with … - 46 -
Figure 4-13. Microblogging example on Foursquare - 47 -
Figure 4-14. Contextual phonebook example in our demo - 48 -
Figure 4-15. Three upper levels of a sample availability status ontology - 49 -
Figure 4-16. Contact details screen in a demo contextual phonebook - 50 -
Figure 4-17. Context sharing preference table of a user Bob - 50 -
Figure 4-18. Notification rules creation and listing screens in the prototype - 53 -
Figure 4-19. Rules exchange between COSMOs and KRAMER - 53 -
Figure 4-20. A simplified part of a situation lattice from a contextual phonebook … - 55 -
Figure 4-21. A Rete network for two sample situations - 56 -
Figure 4-22. Rete network activating with context updates - 58 -
Figure 4-23. Interfaces shared by the KRAMER and a COSMO in their mutual … - 59 -
Figure 4-24. A logical communication between the KRAMER and a COSMO - 60 -
Figure 4-25. A COSMO module connection with the MQTT service - 61 -
Figure 4-26. Message exchange process between COSMO modules and the … - 61 -
Figure 4-27. The main process of the KRAMER server - 62 -
Figure 4-28. Two semantically close situations meaning one generalizing them … - 65 -
Figure 4-29. Two situations differing in structure - 66 -
Figure 4-30. A part of a situation lattice referring to Table 4-1 - 67 -
Figure 4-31. A meta-graph structure for the generalization algorithm - 71 -
Figure 4-32. Cutting empty branches on one tree example - 73 -
Figure 4-33. A sample multidimensional availability status taxonomy - 74 -
Figure 4-34. A prototype screenshot showing options for a suggestion - 75 -
Figure 4-35. A meta-graph structure for the generalization algorithm with … - 76 -
Figure 5-1. Generate random data process diagram - 80 -
Figure 5-2. Time in [ms] dependency for a number of random situations - 82 -
Figure 5-3. Group number dependency for a number of random situations - 82 -
Figure 5-4. Abstracted situations number dependency for a number of random … - 82 -

Figure 5-5. Generate focused data process diagram - 84 -
Figure 5-6. Time in [ms] dependency for a number of focused situations - 85 -
Figure 5-7. Group number dependency for a number of focused situations - 85 -
Figure 5-8. Abstracted situations number dependency for a number of focused … - 86 -
Figure 5-9. Abstracted situations number dependency for different number of … - 89 -
Figure 5-10. Time in [ms] dependency for different number of focus points - 89 -
Figure 5-11. Abstracted situations number dependency for different ratios of … - 89 -
Figure 5-12. Time in [ms] dependency for different ratios of negative rules - 89 -
Figure 5-13. Abstracted situations number dependency for different ratios of leaf… - 89 -
Figure 5-14. Time in [ms] dependency for different ratios of leaf concepts - 89 -
Figure 5-15. Abstracted situations number dependency for different context … - 89 -
Figure 5-16. Time in [ms] dependency for different context dimensions - 89 -
Figure 5-17. Abstracted situations number dependency in the final simulation - 90 -
Figure 5-18. Time in [ms] dependency in the final simulation - 90 -
Figure 5-19. Taxonomy of relations in the test game - 92 -
Figure 5-20. Taxonomy of locations in the test game - 92 -
Figure 5-21. The user test steps - 93 -
Figure 5-22. The game turn summary for one user - 95 -
Figure 5-23. A game suggested notification - 96 -
Figure 5-24. Cross combination of concepts in the generalization process - 100 -

Table 3-1. Our assumptions - 28 -
Table 4-1. Reference notification situations from the described scenario - 36 -
Table 4-2. Pseudocode of the algorithm grouping situations - 69 -
Table 4-3. Situations from Table 4-1 after the grouping process - 70 -
Table 4-4. Pseudocode of the algorithm generalizing situations - 72 -
Table 4-5. Situations from Table 4-3 after the generalization process - 73 -
Table 4-6. Situations from Table 4-3 after receiving a negative feedback - 75 -
Table 4-7. Situations from Table 4-6 after another generalization process - 76 -
Table 4-8. Pseudocode of the algorithm generalizing situations with negative… - 77 -
Table 5-1. Classification of recommender system evaluation properties - 78 -
Table 5-2. A set of default test parameters - 87 -
Table 5-3. A set of final test parameters - 88 -
Table 5-4. Four abstract game rules - 97 -
Table 5-5. Four abstract rules found in the game - 98 -
Table 7-1. Our contributions - 109 -

Glossary

concept

a category of real or abstract objects

context

a set of circumstances accompanying an entity

context awareness

a property of a system being capable of perceiving its context

context dimension

one particular type of context (*e.g.* location, time, activity, speed, *etc.*)

COSMO module

Collaborative Situation Module, a client to the KRAMER system

KRAMER server

a central processing unit in the KRAMER system architecture

KRAMER system

Kind of Reasoning that Abstracts Meta-situations for Empowering Recommendations

lattice

a partially ordered set of objects with unique least upper and greatest lower bounds

microblogging

a practice of posting short messages for social network communities to read

ontology

a structure containing concepts describing universe along with their relations

points of interest

types of situations processed by KRAMER that are likely to be important for people

semantics

the meaning of concepts

situation

everything that is going on in a given time

situation awareness

a property of a system being capable of abstracting its situation

social computing

any type of technological support of a social behaviour

taxonomy

a classification of concepts organized by generalization/specialization relations

taxonomy leaf

a concept in a taxonomy having no specializations

Abstract

Situation awareness and collective intelligence are two technologies used in smart systems. The former renders those systems able to reason upon their abstract knowledge of what is going on. The latter enables them learning and deriving new information from a composition of experiences of their users. In this dissertation we present a doctoral research on an attempt to combine the two in order to obtain, in a collaborative fashion, situation-based rules that the whole community of entities would benefit of sharing. We introduce the KRAMER recommendation system, which we designed and implemented as a solution to the problem of not having decision support tools both situation-aware and collaborative. The system is independent from any domain of application in particular, in other words generic, and we apply its prototype implementation to context-enriched social communication scenario.

Keywords:
situation awareness, collective intelligence, semantic processing, collaborative filtering, rule-based recommendations, conceptual graphs, social computing

Résumé

La sensibilité à la situation et l'intelligence collective, sont deux technologies utilisées dans les systèmes intelligents. La première rend ces systèmes capables de raisonnement sur leur connaissance abstraite sur ce qui se passe. La seconde permet d'apprendre et de dériver de nouvelles informations à partir de la composition d'expériences de leurs utilisateurs. Dans ce mémoire de thèse nous présentons une recherche doctorale qui s'efforce combiner les deux afin d'obtenir, de façon collaborative, un ensemble des règles de situations, dont le partage soit profitable pour une communauté d'entités. Nous introduisons le système de recommandation KRAMER, que nous avons conçu et mis en œuvre comme une solution au problème d'inexistence des outils de support à la fois sensibles à la situation et collaboratifs. Le système étant générique, nous appliquons l'implémentation de son prototype à un scénario de communication sociale enrichie de contexte.

Mots clés :
sensibilité à la situation, intelligence collective, traitement sémantique, filtrage collaboratif, systèmes de recommandation, graphes conceptuels, informatique sociale

Abstrakt

Świadomość sytuacji i inteligencja kolektywna to dwie spośród technologii używanych w systemach inteligentnych. Ta pierwsza sprawia, że systemy te są zdolne do wnioskowania na podstawie swojej abstrakcyjnej wiedzy o tym, co się wokół nich dzieje. Ta druga umożliwia ich uczenie się i wywodzenie nowych informacji na podstawie złożenia doświadczeń swoich użytkowników. W tej rozprawie prezentujemy pracę badawczą nad połączeniem obu technologii, ażeby poprzez współpracę użytkowników otrzymać bazujące na sytuacjach reguły, którymi dzielenie się będzie korzystne dla społeczności jednostek. Przedstawiamy system rekomendacji KRAMER, który zaprojektowaliśmy i zaimplementowaliśmy jako rozwiązanie problemu braku narzędzi wsparcia decyzji jednocześnie świadomych sytuacji i opartych na współpracy. System jest niezależnie od domeny jego zastosowania generyczny, a jego prototyp zaadoptowaliśmy do realiów komunikacji społecznej.

Słowa kluczowe:
świadomość sytuacji, inteligencja kolektywna, przetwarzanie semantyczne,
rekomendacje bazujące na regułach, grafy koncepcyjne

1. Problem statement

The aim of interpersonal communication services is to support and enhance the human interactions as they exist in nature. For example, the very telephony extended the distance for natural human conversations making it much more convenient to initiate. Today's technical capabilities are much more sophisticated than when the telephone system was introduced. Phones have been made cordless and embedded with multiple sensors. As a result, they have become users' smart personal assistants. The problem extensively explored lately is how to exploit this device intelligence to further facilitate human natural and social behaviours.

In our research we look into the fact that mobile users' activity, *i.e.* smartphone users' activity, generates tons of data, which enables distinguishing traces of their ever-changing situation. For once, modern personal devices are capable of sensing their users' whereabouts or any context in general. Moreover, they can also remember any user interactions with them. For instance, in case of mobile phones, any decision on social interactions is stored in a call log (synchronous communication) or in a message box (asynchronous communication). This stands for a lot of unstructured data about phone users from the device perspective only.

There are many works to uncover the structured knowledge from user context traces, for example learning daily movement routines [ZHE09], or learning tasks usually performed in a mobile environment [LEE10]. The common approach is to apply pattern or association rule mining algorithms to extract the corresponding structures of contextual data. As a result, designers may be provided with interesting insight into the possible enhancements to their service usage. An alternative would be giving users themselves some additional information regarding their current situation, especially if it is not frequent that a particular one occurs.

In case of critical situations bearing some important circumstances that one has not encountered before, some help from a smart assistant could be of great value for making a respective decision. In classical human interactions, people exchange their experiences by asking questions and sharing advices with one another. This process can be supported by technology, given the vast amount of above-mentioned activity data. This group experience remains to be explored for identifying important situations, and suggesting solutions in a collaborative fashion among users.

Following [SUR05], under the right circumstances, groups are remarkably intelligent, and are often smarter than the smartest people in them. Those circumstances are appropriate reasoning mechanisms, deriving their intelligence from users' participation in the information shaping process. Aggregated and further recombined data transforms collected intelligence into a truly collective intelligence [GRU07]. According to [ALA08], when a group of individuals collaborate or compete with each other, intelligence that otherwise didn't exist suddenly emerges. This intelligence is computed by knowledge mining algorithms. As a result, such interactively derived social knowledge can be shared with all community members for their benefit via recommendation technologies, reputation systems and other decision support mechanisms.

To the best of our knowledge, applying the same principle of a collaborative system to the user situation awareness has not yet been well explored, if approached at all. We argue that while people naturally learn from one another about the importance of situations needing their reaction, modern devices may provide enough valuable context data to propose a system that supports the sharing process. Being provided with information concerning other people taking often one kind of action in similar situations that one encounters for the first time may result in a set of new services of empowering social value. This thesis proposes a system enabling such services. Its codename is KRAMER, Kind of Reasoning that Abstracts Meta-situations for Empowering Recommendations. We predict that it might become a novel smart decision support tool.

The rest of the document is organized as follows. In Section 2 we describe the state of the art within the research areas of situation-awareness and decision support systems. Analysis of multiple research activities reveal them not to be dealing with the problem as defined above. Therefore, we propose some assumptions and functional requirements for a system to constitute a good solution in Section 3. Discussion how it enhances the current research state is also provided. The next section brings the initial ideas closer to their realization. After having chosen a domain of application for our system, we proceed to details on the architecture and implementation of all functional blocks. This is all presented in Section 4. The technologies and mechanisms selected ought to fulfil all the prior assumptions, which we verify through tests in Section 5. Afterwards we abstract the system to be applicable to multiple domains and we present

a couple of them in Section 6. Finally, we conclude and point out some future research directions in Section 7.

2. Theoretical background

In our research we seek to exploit collective intelligence in the domain of context and situation awareness. Systems based on this kind of intelligence make their individual users benefit from knowledge of more experienced ones. In that sense, such systems create a community of users, which mutually contribute to its growth, and in turn get a support for areas that they are no experts of. Adapting this principle to context would require operating on experiences of particular sets of context values, which may be interpreted as situations. Furthermore, one would expect a collectively situation-aware system to support making decisions while encountering new but important situations.

For this reason, in this Section we look into the existing research works concerning the areas of situation awareness and decision support systems. We investigate if there are already existing solutions allowing communities to gain collaborative awareness of important situations. We explore equally different approaches for defining situations for them to be taken into account in smart systems, as well as different technical solutions for such systems to empower decision making. This state of the today's art leads us to introducing our system in the next sections of this dissertation.

2.1. Situation cognition

In this subsection we investigate the state of the art on the ways that situations are seen in technology. In particular we discuss definitions of context and situations provided by both encyclopaedic entries and the Situation Theory, show different approaches to model real life situations in artificial systems, and explain how incorporating situation awareness makes those systems become intelligent. In our study we focus on finding a representation for context, which would form situations well understandable by humans. In fact, we seek for it to adapt the meaning of a situation from one agent to another, making it a little blurred, which results in having a mathematical definition rigor a lesser priority. Therefore, we are referring for the remainder of this section to contextual situations, which we expect to be at the same time expressive for users and manageable by a system.

Finally, we search for any situation-aware systems that already provide any type of decision making support, especially based on a collective knowledge paradigm.

2.1.1. Situation Theory

Our interest in systems that are aware of their situation and of the situations of their users implies the need to define what a situation is. In encyclopaedic entries including Cambridge Advanced Learner's Dictionary & Thesaurus[1], The American Heritage Dictionary of the English Language[2], and Collins English Dictionary[3] we see the term "situation" to relate to a set of happening things, existing conditions, circumstances and surroundings of something, someone, or other point in the time-space continuum, like an event. Other languages (Słownik Języka Polskiego PWN[4], Larousse[5]) also tend to place a situation in relation between an entity of any type and some surrounding conditions.

What is therefore a difference between a situation and a context? The same set of dictionaries explains the context as the circumstances of an event. This makes the two terms very close, with only a slightly broader meaning for the situation. Cambridge Dictionary makes the relation explicit as the context is to be a situation within which something happens. It is then of no surprise that the two terms are frequently used in a synonymic fashion. They are further fused in spoken language, which is known to be less accurate than the written one [MEC07].

To find some distinguishing features between situation and context we look into the domain of event processing. Authors of [ETZ10] point out that situations are things that happen, and which have a meaning, whereas context is a state, an area in an abstract space defining some conditions. Based on this approach, and the initial use of the term "context awareness" with relation to sensor technology, we derive our working relation between the two terms in question. We treat situations as meaningful, possibly semantic, interpretations of some context setups.

In our quest to capture the idea of a situation we refer further to the theory formulated by Barwise and Perry, which tackles the problem from a point of view of such disciplines as cognitive science, computer science, linguistics, logic, philosophy, and mathematics [MEC07]. The situation theory [BAR83] is an effort to capture the meaning and semantics of situations, by approaching them from both mathematical rigor and practicality perspectives. Even though the theory is considered as not-well-

[1] http://dictionary.cambridge.org/
[2] http://www.ahdictionary.com/
[3] http://www.collinsdictionary.com/dictionary/english
[4] http://sjp.pwn.pl/
[5] http://www.larousse.fr/dictionnaires/francais

founded set theory [ACZ88], it helps understanding the relations between a situation and information, and between different classes of situations.

In [COO91] a situation is presented as an object in situation theory related to the collection of ***infons*** that supports it. An infon is another concept coming from the theory, which is an object carrying information. It is defined as a relation of its arguments with either positive or negative polarity <<R, a1, ..., an, 0|1>>. A piece of information is represented in mathematical terms capturing a number of elements being in a relation to be either true or false. Therefore, a situation is supported by the composition of a number of such pieces of information. It is not, however, defined by it.

In order to capture further the meaning of situations, the theory explores a more philosophical approach. In consequence a situation is said to simply describe "what is going on". However, no ultimate definition can be provided. Devlin explains that situations are abstract objects, elementary concepts, which cannot be defined precisely using other mathematical items [DEV91]. In that sense a situation is an abstraction capturing the state of the universe relating to the complete set of infons supporting it, mixed with a meaning.

Fortunately, as human beings are able to perceive their situations, and reason upon them, so can artificial agents. However, neither people nor systems are likely to capture the state of the whole universe, which finds its reflection in the definition of ***individuated situation***. Devlin explains individuating a situation by an agent as perceiving only a part of the reality that is necessary for a situation to be picked out [DEV91]. An individuated situation becomes a relevant subset of the state of the universe [DEY00a]. This implies a limitation to a finite set of facts describing situations, and more importantly enables modelling them for representing in smart systems.

2.1.2. Situation models

In order to reason upon situations, agents and systems need to structure their understanding of what is going on at a moment. This calls for a model to represent situations within such computing entities. Those representations are sure to capture but a fraction of a real situation, with respect to its understanding in a situation theory. In this subsection we give an overview on the context and situation models. We discuss their differences regarding their expressiveness, ways of associating a meaning and technical implications.

[STR04] gives a complete survey on the context model types, which include the key-value pairs, logic notations and ontologies. It provides also mark-up schemes and graphical models as ways of rendering context knowledge processable by machines and humans respectively. The authors provide equally a set of 6 model evaluation parameters, and they ascertain that the ontology-based models satisfy all of them best. It seems, for example, that ontologies enable context reasoning and facts derivation, as explained in [WAN04], and dealing with uncertainty [GU09].

Situations, however, even if being a limited part of the world [BAR83], can represent something more abstract than a value for just one context dimension. Therefore, authors of [ANA06a] model their situations as somehow abstract concepts, which are taken from a situation taxonomy. A set of abstract labels is arranged in the way that "formal meetings" are special cases for "meeting", for example. In this model, the situation concepts, and the whole taxonomy for that matter, are set apart from the context and the corresponding context ontology.

This approach seems too simplistic and not expressive enough. Placing every meaningful situation in one taxonomy is too limiting, as any sibling concepts should be disjoint, and parent-child relations require a strict generalization-specialization of corresponding meanings. There exists a possibility of making the model slightly more sophisticated, should the multipleinheritance be introduced to the taxonomy [MAR02]. In that case two child concepts of another one could have orthogonal meaning rather than complementary (*e.g.* "business meeting" and "stand-up meeting" are two descendants of a "meeting", which do not exclude one another).

Another problem with the presented model is the lack of connection between the situation and context ontology structures, preventing any translation from one to another. The necessary link is discussed in [YAU06]. The situation ontology is divided into two logical layers, one for lower level context concepts, and another for more abstract situations. It is an upper level ontology that includes a relation between an atomic situation and a context value element for a context dimension concept.

The authors of [ANA06a] try to deal with the same issue by introducing a new situation model in [ANA06b] that considers situations as compositions of different context dimensions. They limit those dimensions to the following four: spatial, temporal artefact and personal. The situation model here becomes more expressive, carrying more information, for the fact of being composed of several pieces of information.

Those pieces of information assembled to represent a situation are called ***characteristic features*** in [MEI04].

Composing a situation out of several elements is present also in other context modelling approaches. In situation theory, the logic notation introduces basic infons representing simple situation as in [AKM96], which can be further combined to form compound infons [KOK09]. In [COO91] authors state that situations are in general a collection of infons that support them. An abstract situation, which can be sometimes expressed in few words, is very likely to be explained by giving several facts that do and do not accompany it.

In [PAD04] we see yet another interesting approach to model this composed nature of situations. The authors adapt a graphical representation of context within n-dimensional spaces, where ***n*** is a number of context dimensions considered. Whereas a point in that space represents a precise reading of context sensors, situations take form of subspaces. This makes a situation to be a composition of those context dimensions, which it spans, and not those for which they are completely flat (of length 0). Furthermore, the size of the span determines the range of context values for a given situation. A similar adaptation can be made for the vector context representation approach in [DEL12].

The previous space-based situation model does not consider any situation semantics whatsoever, but the presented principle of seeing a situation as a Cartesian product of several context dimensions has been considered in [KNO08]. Should every context dimension of a situation be considered and modelled separately, their combined Cartesian product would constitute a valid representation of a situation. Other researchers deal with such one dimension representations, making them based on <type, operator, value> triples [YAU06], which should describe an entity (called *substantial*) [COS06].

Should a set of simple, one dimensional semantic-rich situations, which follow the above recommendations (*e.g.* "humidity is 77%" or "wife is in a kitchen") be put together by adding "AND" operators, one would simply receive composed sentences, possibly quite long ones. In order for them to be processable by agents one would expect better structured representation. We identify conceptual graphs as able to represent quite complex sentences and logical expressions [SOW83]. Using them as a model of contextual situations is in return quite expressive.

It also remains to be verified if conceptual graphs representing situations are manageable by systems, and if they enable efficient reasoning upon them. It is often of interest to compare different situations and judge their mutual similarity. In the case of a context space-based model, comparison is a matter of a distance between subareas [PAD04]. Similarity of situations can be also measured in terms of a distance in ontologies [GAN08]. Conceptual graphs, having multiple ontology concepts incorporated, is just a more complex case of the latter.

From the theoretical point of view, the six conceptual graph operators [SOW08] may make one graph to project on another. Thus, a network of dependencies between many more graphs can be created. As far as a more technical approach is concerned, there exists multiple implementation works on conceptual graphs comparison [MON00] [MON01] [POO95] [ZHO02] [REE05]. Even whole ontologies may be compared and matched with one another for that matter [CRO07]. The computational complexity reported in the reports is encouraging. Our adaptation, and extension of the conceptual graph model for situations is detailed in Section 4.2.

2.1.3. Situation awareness

Situation awareness is a property with a crucial impact on decision making and performance of both human and artificial systems [END95]. In his situation awareness theory introduction, Endsley presents a model of situation-awareness as an ability to perceive surrounding elements, comprehend their meaning and project their statuses into the near future. The author argues for that ability to require a much more advanced level of understanding than just being aware of numerous pieces of data. What separates it from context-awareness is operating on already abstract semantic interpretations of context, which makes it more stable, more certain, and, most of all, more meaningful to agents and systems [YE07].

The use cases mentioned in [END95] are complex decisions systems like those in aircrafts or air traffic control. Such tactical and strategic applications, which take into account many circumstances and have an enormous impact on the safety of people, are already being based on situation-awareness technology. Also some systems more, let's say, down to earth, recognise this technology as key for decision support in: information logistics [MEI04], network communication [BEI03], or driver assistance [MCC07]. All these systems introduce intelligence in a form of appropriate reaction of any type for given situations.

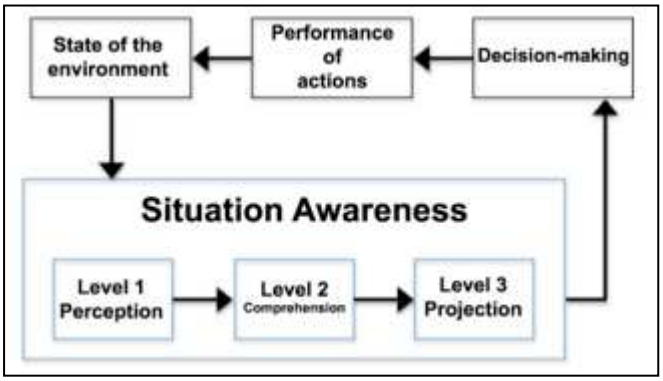

Figure 2-1. Endsley's model of situation awareness [source [END95]]

However, all of those systems are based on rules predefined for the system, or at best those introduced by a user while initiating the system. While it is probably the best solution for systems needing strong reliability, like the ones responsible for the safety of people, other everyday decision support applications could provide a little more customization and help for doing so. We mention in the introduction section that defining a functionality of a situation-aware system could be a collaborative effort, making the system's intelligence become collective. None of the systems reviewed here introduces such possibility.

2.2. Decision support

In this subsection we investigate the state of the art on different systems that support their users' decision making. In particular we focus on recommender systems, especially those incorporating context awareness, and collective knowledge. We search for any application of such systems in the domain of situation awareness, and any indication as to what would be the profit of doing so. We point out those technical solutions that could be adapted in introducing a collaborative and situation-aware system.

2.2.1. Recommender systems

The way that people help each other in making decisions is by giving advice. A system replacing humans in this kind of support is called a recommendation system. A transcription of using knowledge about the preferences of a person addressed with a recommendation is used in so-called content-based recommender systems. Suggestions are made there if a description of an item seems to correspond to a profile of the user's interests [PAZ07]. The items in question can concern multiple domains ranging from items to buy to things to see, places to visit, *etc.*

There are multiple technical approaches to render giving suggestions by computing systems possible. If an item's characteristics are given by a web store, for instance, a system still needs to gather information concerning user's preferences. This is done either by asking the user herself to provide the necessary pieces of information, or by inferring it from the history of user's interactions with the system [PAZ07]. Both solutions have their flaws. The first requires an effort from users, which they may be not willing to make, while the second takes time for an algorithm to learn. In either case having to much or to little knowledge may result in providing too few or too many suggestions respectively.

In [ZAI02] the author suggests for simpler cases to limit the amount of knowledge gathered on a user and rather compare the amount already there with association rules mined from the whole user population. This approach is based on a collective history, making the learning process faster. The condition is that for a given domain there exist few global trends that all users are very likely to follow. In this case the recommendation system is applied to e-learning, and following particular learning modules is predicted on those few already done.

On the other hand, [CAO06] presents a domain, where quantitative history data would not prove any useful. Electronic products are items that become quickly obsolete and in the same time frequently introduce completely new technologies. Not only can the history purchases be incomparable with a current market offer, but users may have little knowledge of what the offer is. Therefore, instead of asking for a desired value of graphic card memory, the presented system asks users what is the importance of playing games and graphics design to them. The suggestions are further supported by domain experts and fuzzy logics.

In spite of different approaches to learning user interests models, content-based recommender systems are limited by the information at disposal for both items and users, *i.e.* the content. Providing this data in a manual way imposes a considerable effort, while obtaining it automatically is not always possible. Moreover, the classical recommendation system approach does not consider the fact that the very users tend to change taste in function of multiple factors. This is an especially crucial point in context and situation awareness, where decisions are taken in response to particular situations, and not in all of them.

2.2.2. Contextual systems

Context awareness is a property enabling considering circumstances of events and surroundings of an entity, see Section 2.1. Applying it to the decision support systems leads to context-aware recommender systems. Such systems incorporate into recommendation ratings a third dimension, the context, to the usual two: user and item [ADO11]. The interest of a user in a particular item may vary in function of different context dimensions, like mood, time of day, fatigue, *etc.* For that reason, the suggestions can be filtered to match better preferences in a given context.

[ADO11] presents three patterns that the use of context can adapt in recommender systems. The first two include suggestion results filtering either before or after the classical recommendation derivation is performed. Those two approaches exploit proximity between contextual data and validity context defined for user interests [NAU10]. The third approach requires incorporating context into recommendation functions. Either way, the context component is nothing but a parameter to augment the accuracy of the suggestions made for any type of item. There is no reference whatsoever to recommender engines which would consider context as such an item *per se*.

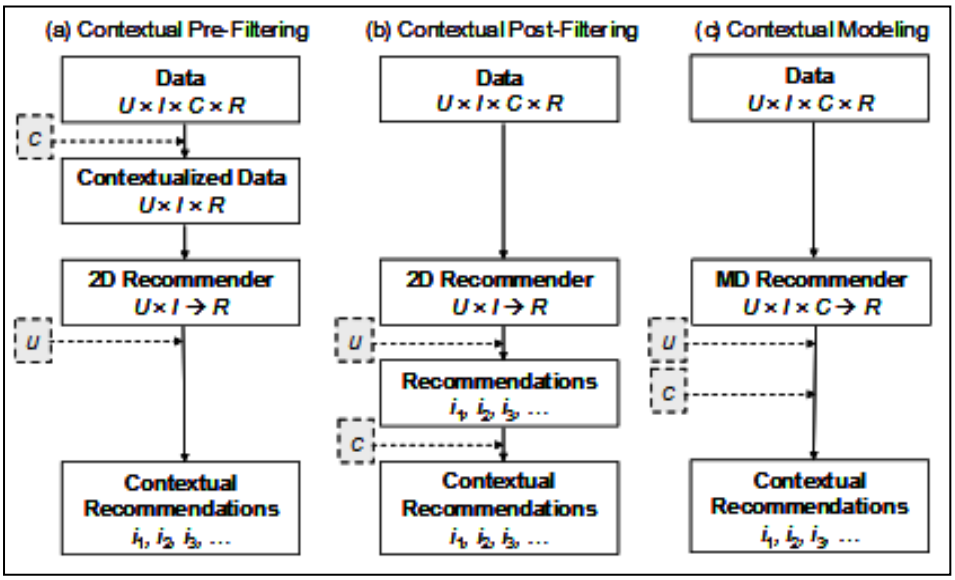

Figure 2-2. Paradigms for incorporating context in recommender systems [source [ADO11]]

To connect the context with particular item preferences, the corresponding association rules can be mined like in [HON08]. The links can also be established using the collaborative filtering technique, like in [CHE05]. In the latter case a prediction for a user preference for an item depends on the item evaluations in similar contexts, whereas similarity between contexts is measured by similarity of evaluations for the same objects in those contexts. Therefore, whatever the context dimensions values really

are, situations are considered similar if a preference for a majority of items varies a little.

Even though [CHE05] refers to collaborative filtering, the notion of collectiveness and cooperation is missing there. Whereas the technique seems to relate to an algorithm applied usually to find similar, like-minded users. In fact, in the context-based recommender system survey [PAZ07] authors draw a conclusion that combining content and collaborative information can supplement suggestions by the opinions of a community of users. This is also an approach we aim to explore. However, neither of the context-aware recommendation systems studied make use of collective intelligence.

2.2.3. Cooperative systems

As discussed in the introductory section, the collective intelligence principle may enrich systems, both social and computing ones, with information not available directly [SUR05] [ALA08]. This knowledge is best considered in decision support systems. As in real life, a more abstract understanding of the world comes from observing individuals collaborating and competing. Patterns of their preferences and choices do not exist unless observed in a larger scale. And those patterns constitute a social experience, the concept of which can be exploited in recommendation systems.

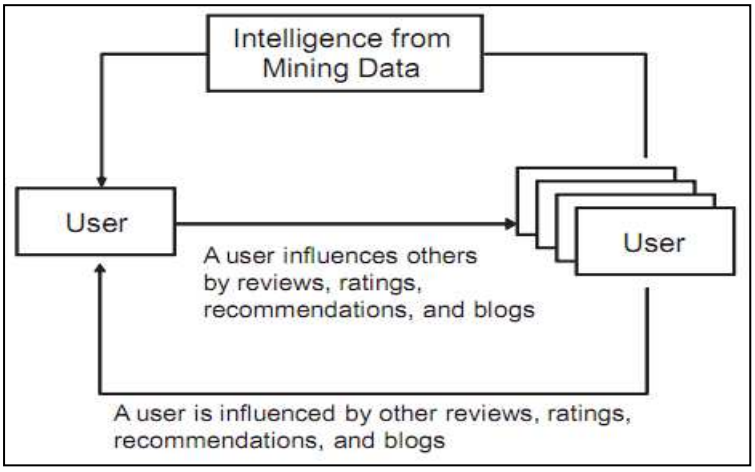

Figure 2-3. User influenced by others directly or through derived intelligence [source: [ALA08]]

There are two sides for such collaborative systems. First, they transform sets of their users into true virtual communities [HIL95]. Those communities do not constitute virtual reality communities, as members do not interact directly with one another. Still, the very nature of those systems makes the experience to be communautary. Second, the community experience, or any of its user data for that matter, is not only stored and redistributed. New data, involving meanings and conclusions, is abstracted, making ***collected intelligence*** to become ***collective intelligence*** [GRU06].

One of the most popular cooperative techniques adapted by recommender systems is collaborative filtering. In a general sense it stands for filtering information as an effect of any agent collaboration. In particular it can be implemented as a statistical tool to predict one's preference towards an item, which is based on a database of preferences for different items by all users [SAR01]. It has been found especially successful in e-commerce [SCH01a], where web stores measure the utility of collaborative filtering techniques in a growth of sales thanks to "other users who bought this product were also interested in..." lists.

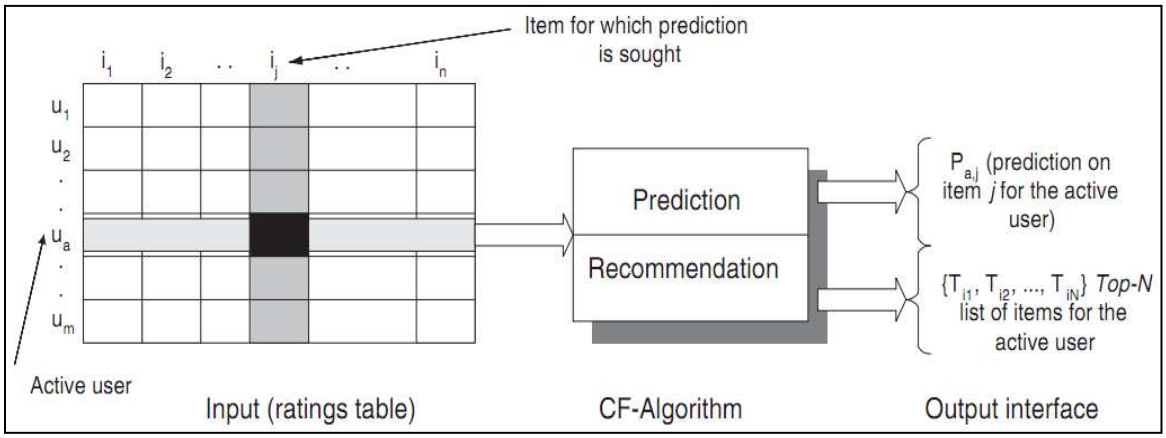

Figure 2-4. The collaborative filtering process [source [SAR01]]

According to [DUC09], collaborative filtering is most useful for people unfamiliar with a given area. In fact, experts rarely need suggestions coming from systems that may be sometimes wrong. Inexperienced users, however, frequently need any kind of help to quickly find an item of interest among thousands of not interesting ones. What makes the collaborative approach particularly interesting is the fact that the recommendations do not come from a biased seller, not even from the majority of other users, with which one may not necessarily identify. *A priori*, collaborative filtering presents items that like-minded users, which are those whose preferences match in many cases, find interesting.

As far as using context in recommender systems, there is no notion of such user similarity in the context-aware collaborative filtering system in [CHE05]. We find it in [MUN10], where recommendations employing collaborative techniques depend on two context dimensions: location and time. However, the system presented there seems to only filter suggestion objects based on relevance to recommendation requests (movie suggested needs to be one of those played the evening in the given cinema). In consequence, the context considered is a recommendation context, and not a user-dependent one.

3. Proposed solution

In the current state of the art we have found no documented research works on technology empowering user context and situation awareness with an experience of other people. Context is treated rather as a set of circumstances to filter derived suggestions, for instance in [MUN10]. We argue that a situation itself can be a subject of a recommendation. Encountering one could lead towards a wider range of actions and decisions for a user to take than just buying an item or not. In the same time we recognise that collaborative techniques, especially the simplicity with which statistical users-items relations are transformed into collective intelligence are very promising for our system. We aim to make their application in the global situation awareness domain more proactive, like the works in [HON08] were with respect to the local context history.

In the following subsections we define the goals and the frame of our solution to the problem. We analyse also the main functionalities to provide, which will lead to design and implement the solution in Section 4.

3.1. Working hypothesis

As stated in Section 1, we seek in our research to investigate a possibility to apply data mining techniques onto user data of contextual nature. As a result, we expect to harvest new knowledge about user behaviour in particular situations. We are especially interested in exploiting the collaborative filtering principle in user telecommunication interactions. We believe that it could lead in turn to providing new intelligent social communication services, where each user would benefit from the experience of other users in certain contextual circumstances.

We design a system that makes the idea of being aware of situations a subject of a recommendation. This approach has not yet been explored in the research works that we are aware of. The previous section summarizing the state of the art shows no connection whatsoever between the systems cognizing context and situations, and those providing recommendations, even if the latter may take into account some context notions. Finally, we focus on obtaining a seamless connection between the two areas by applying collaborative, social techniques.

At the very beginning of our research we formulate the following three assumptions driving this thesis:

Table 3-1. Our assumptions

A1.	*Collaborative filtering techniques applied on context data would create new knowledge*
A2.	*This knowledge could introduce a range of new interpersonal communication services*
A3.	*Semantic modelling and reasoning should be employed for context data manipulation*

3.2. Informed decisions

The previous subsection lists several hypotheses for our research. Should we treat them as functional goals of a solution to the problem given in Section 1, we get a set of requirements for our collaborative situation-aware system. Namely, it ought to present its users with situations filtered from activity traces of the community in order to empower both their social situation awareness and decisions related to their interpersonal relations. The main object being manipulated therefore is the situation, while the principal technique used is collaborative filtering in its general sense.

Exchanging experience with one another in interpersonal interactions is called giving advices and suggestions. Smart systems that can do the same are called recommendation systems. A subset of them is known to implement collaborative filtering algorithms, which extract knowledge and experience of a group of its users in order to share it with other ones. We choose for our system KRAMER to be a recommender system that suggests taking an action whenever the current context state has been found relevant by other users in the past. This way, one would learn from the experience of others and would not miss the right circumstances to react to accordingly.

Figure 3-1. The KRAMER system transferring an experience of a group to a user

A particular state of multiple context dimensions can be described in one word as a situation. One's situation is a set of circumstances, of all what is going on around [BAR83]. In practical terms it is a processed, abstracted context [DEV91] giving its semantic interpretation [YE07]. And such situations are objects of analysis for us. Finding important ones requires the system to observe patterns in users activity decisions in similar circumstances. If a situation inspires taking certain actions by

people, it is a sign that suggesting taking an action could be beneficial for someone who has no experience with similar situations.

Now, situations are not only simple single concept ones (*e.g.* running, at a meeting). They can be quite complex composited beings describing a state of several entities in their social context. For example, let's consider a situation, where one's daughter has finished her classes while his wife is busy at a meeting. First, it is definitely a situation composed of a couple of circumstances, which we call context dimensions. Second, it might be important for someone, whose wife is usually responsible for driving their daughter home - this might require an action to call his daughter to wait for a while or to simply go pick her up himself.

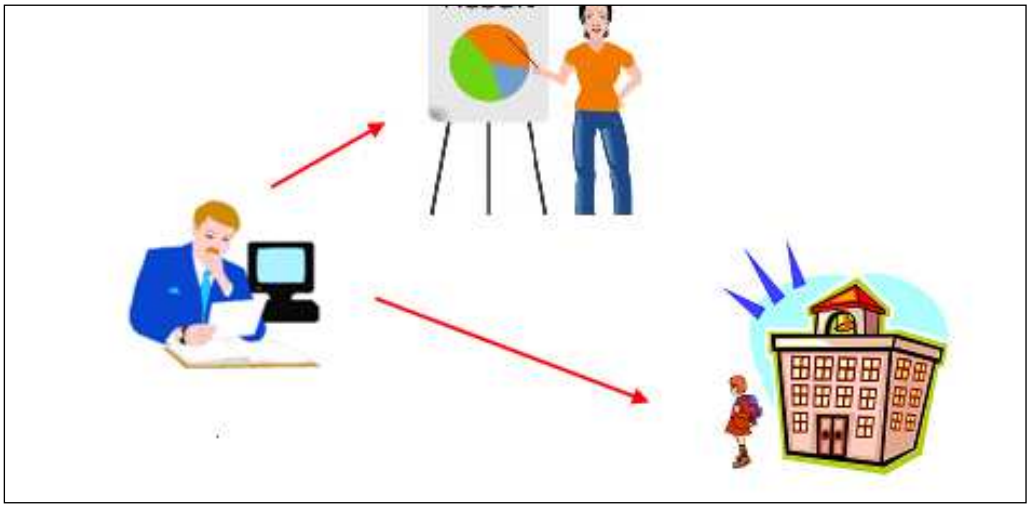

Figure 3-2. A situation of a user is a composition of his relatives and their context

As a consequence of having situations more semantically complex than object references that collaborative filtering mechanisms normally deal with, our recommender system should provide an adjusted implementation of its algorithms. Typical statistical methods used for suggesting books or movies in web stores are based on measuring similarities between those objects in terms of their shared owners with similar tastes. In the case of the situations, having the exact same one happening for multiple users depend on the degree of granularity of context [BAL07] values. Therefore, it might be necessary to include a semantic similarity measure.

Besides, the standard mechanisms of processing the objects to find those popular ones for presenting the respective suggestions to users should be present. Furthermore, users given with a notification of an important situation should be able to quickly evaluate it by indicating, for example, that they indeed find it interesting enough to be presented with it every time in similar circumstances, or on the contrary, that they do

not wish to be bothered with it again. That decision should be taken into consideration by the system, causing the importance of situations to be constantly revaluated.

As our research for this thesis focuses on creating a system applying collaborative techniques on identifying important situations, it does intentionally avoid the problem of harvesting those situations from raw context data. Finding a set of context dimension values associated with frequently fired action is a subject of data mining algorithms that would analyse vast logs of community activity. They could constitute an entry sub-module for the KRAMER system. We make the assumption, however, that there exists a way for users to contribute with the situations they find initially important. Then our social mechanisms can make those situations be shared.

3.3. Functional decomposition

In order to learn building blocks needing an implementation, and eventually to propose an adequate architecture of the KRAMER system, we look into its reduction to the most important functional components. In the previous subsection we decided that people can be collaboratively aware of important situations via suggestion mechanisms, which makes KRAMER a recommendation system. Therefore, its abilities to both manipulate situations and provide suggestions are two main tasks of our system. This is expressed on the second upper level in Figure 3-3.

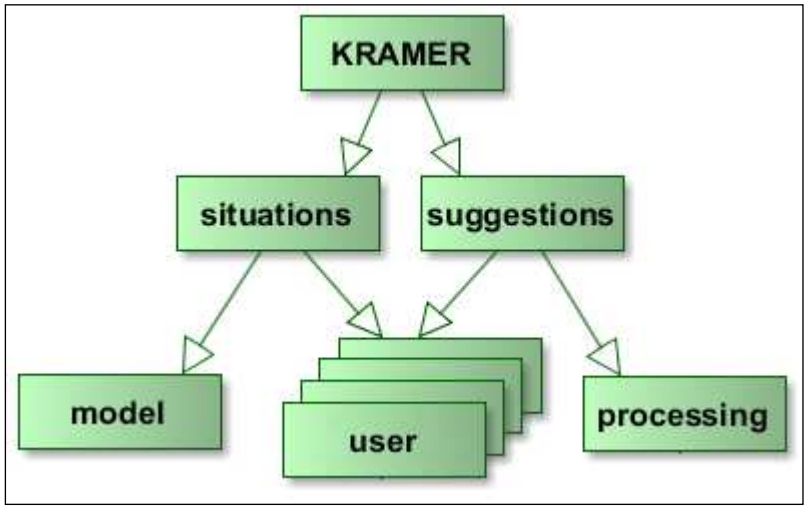

Figure 3-3. Functional decomposition of the KRAMER system

Digging deeper, we notice that suggestions are an effect of some situations processing algorithms, which harvest those important ones. From another perspective, recommendations have a purpose of being provided to a user. One might say that they require such recipients to exist at all. It is also those users who provide the situations to the system via their context-sensing electronic devices. Finally, the situations need a

well defined model to enable their processing by the system. The "model", "processing" and set of "user" modules are on the third upper level in Figure 3-3.

Let's focus on the set of blocks representing users. For each user to be able to profit from the suggestions, she needs to have an interface with the system. This connection should be available via a device that could sense the user's context, and exchange information about situations with other users, like family members or friends. At the same time the device ought to be personal enough to follow its user in everyday activities, while preferably not being shared with others. For all of those reasons, we propose the device to be a mobile smartphone. It is personal [RAE05] and new models are capable of sensing more and more context dimensions. Furthermore, connectivity for context exchange can be maintained among contacts in the phonebook.

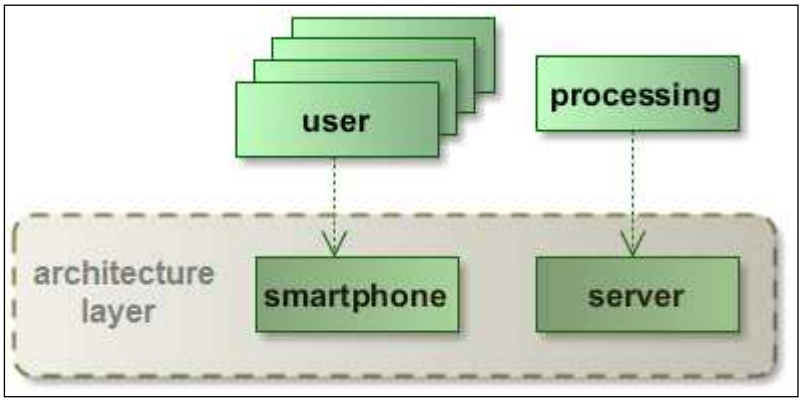

Figure 3-4. Client-server architecture inferred from the functional blocks

Each user could browse through her contacts to learn their situations, and might define complex situations to be notified of on top of that. Those locally important situations should then be contributed to the system, which would perform its processing in one central entity, a KRAMER server. The situation model would be obviously common for all user clients and for the server. As a result, a client-server architecture emerges. We present it along with the communication protocol between entities in Sections 4.3 and 4.5 respectively. We discuss the particular functional blocks in Section 4.2, 4.4, 4.6 and 4.7.

4. System implementation

While the previous section introduced a functional overview of the KRAMER system, this section is focused on bringing all implementation details of each of the system's modules. We start by choosing as a reference several use cases on one specific domain, which would demonstrate best the functionality of the system. Then we propose a data model for representing situations and a global architecture of the system. Finally, we discuss each system module separately along with communication mechanisms in between. The heart of novelty of the KRAMER system is presented mainly in subsections 4.2 and 4.7.

4.1. Chosen domain

An implementation of a generic collaborative situation-aware system prototype needs selecting at first one specific domain of application. With our system we aim at introducing a new set of interpersonal telecommunication services, which determines the domain. In the rest of the subsection we explain how the solution is applied to the domain, as well as for whom it is addressed, and we give a sample scenario with one use case for future reference.

4.1.1. Interpersonal relations

Situation-awareness is a property of perceiving and understanding the circumstances in order to evaluate them and to make conclusions. It is a property of human beings and autonomous intelligent systems used to support their decision making. In our research we are more interested in the former, that is humans and their social behaviours. By enhancing their awareness of current situations we aim at introducing a new set of interpersonal services supported by our system, KRAMER.

The Section 3.3 sketches a general decomposition for functional blocks of the KRAMER system. It mentions a central logic along with numerous satellite elements, which are responsible for two main tasks. First, they gather information about users' situations. Second, they enable maintaining social relations between users based on this information. Therefore, in practice those would need to be some personal tools with communication capabilities.

For that reason, we have chosen a mobile phone as an interface between a user and the system. Today's smartphones are able not only to place calls or engage an asynchronous data communication, but also to sense the environment (temperature,

light, noise, *etc.*) of a user. They assure at once both the communication and situation-aware aspects. In the same time, such devices are frequently carried around by owners and hence may be considered as very personal [RAE05].

Therefore, we have decided to express the situation-enabled interpersonal relations in form of a contextual phonebook application on a mobile phone. As explained in greater details in Section 4.4, such a phonebook is a live contact list enriched with each contact's current context (availability, location, *etc.*). While smartphone sensors can observe the environment, their communication channels support sharing that information with others on a preferred level of details.

We have discussed in one of our publications the social mechanisms accompanying a context-enriched phonebook. It appears according to other researchers (*e.g.* [SCH01b], [RAE05], [BAR08]) that presenting a context of each participant of a conversation, otherwise available in classical face-to-face relations, enhances artificial services with a notion of naturalness. Furthermore, exchanging those situation dimensions between users forms a base, on top of which the KRAMER system may operate.

Figure 4-1. Phonebook 2.0 Android application screen [source: Google play]

In consequence, the situations that KRAMER provides suggestions for may be composed of several context dimensions of a given user and her close contacts. Those dimensions are the same with the ones presented in a phonebook application, which we limited for the needs of the prototype to two such dimensions, user's availability and

location. Situations discovered by the system as important are therefore those that share contacts involved along with their local situations.

Naturally, interpersonal relations in telecommunication services are just one choice for an application domain. It matches well with the system requirements and it keeps the system directly evaluable in terms of usability, which we exploit in the user test, see Section 5.2. However, KRAMER is a tool generic enough to be applied to other domains, which we discuss in Section 6.

4.1.2. Target group

KRAMER is a recommender system, which first discovers situations important for a meaningful part of a user community, and then makes suggestions on actions to take whenever they occur for any community member. Actions that the community members take in particular circumstances are learnt by the system, and transformed into situation-based recommendation rules. In that sense KRAMER is a rule-based system.

Systems based on rules are frequently used to gather an expert knowledge necessary to either take optimal and fast decisions, or to support such decisions of other human experts [HAY85]. This is not the case here, as KRAMER gains its experience in a collaborative fashion from ordinary people and their everyday behaviour. We might say that the "expert" knowledge is elaborated from a mass of non-expert personal decisions.

In the real life community members share their experience with one another for the profit of those less experienced and for the progress of the whole group. The KRAMER system supports this social mechanism. In consequence, the target of suggestions provided by the system would be an ordinary member of the community, rather than any domain expert. Furthermore, KRAMER encourages also its consumers to revaluate the rules so that the system can improve in time. In that sense, the collaborative knowledge gathered from non-expert community members transforms the community into one collective expert in the given domain.

4.1.3. Use cases

Let's consider the following story to be an informal introduction of some use cases for the KRAMER system in the social computing domain.

Imagine Ed, a middle-class engineer, privately a husband and a father to a girl, and deep inside a devoted football fan. A couple of weeks ago he got his first smartphone. He was encouraged by a number of context sensors embedded in the phone and necessary software, which promised its intelligent self-adaptation to his needs. In fact, Ed was very sceptic towards the technology lately, which seemed to add unnecessary burdens upon users, rather than to facilitate natural ways of contacting his friends, for example. 'If the telephony is supposed to bring people together, why don't I get the impression of seeing my contact's status or location', he used to say. His friends have frequently provoked him laughing, 'You'd like to know everything about everyone, don't you?' 'At least, I wouldn't disturb you with my calls when you're taking a shower and I wouldn't get irritated when you don't reply in that situation', he always replied.

But as soon as he turned on his smartphone, he got pleasantly surprised. The basic phone services, like placing a call or sending a short message, were available from a context phonebook. There were already several of his contacts using this technology, some of them were sharing their dynamic context information with him, which he saw next to their names and photos. 'Finally I can learn of my friends' availability before I call', he cried happily. 'Honey, did you know your sister is abroad again? Calling her for some chit-chat will cost her too, you know? I shall enable visibility of my status as well. Hey, how does it know that I'm riding in a train - probably it scans my agenda or senses my activity. I will not share such details with everyone. But I shall with you, Hon, so that you knew when I'm in the meeting not to be disturbed...'

Ed's wife was already used to his jokes but couldn't resist not teasing back. 'I know you're very busy not to be disturbed office coffee drinker, but don't forget to sometimes drive your daughter home when I'm stuck at a meeting, huh?' But Ed had read his phone manual a little further by that time and he had a response. 'There is no worry, once we get both you and our little girl a couple of smartphones, I can program a rule that whenever she's finishing classes while you're at a meeting, I would get notified by the system! Isn't that just splendid?' As Ed said, so they did indeed. With one simple rule written via the same application, Ed assured his daughter won't loose time waiting in school because of his short memory.

One evening, after having his daughter driven home, he decided to benefit from the fact of his wife still being held at work - he turned on a TV with a football match just starting. The very same moment his phone buzzed. This time it wasn't a simple notification to one of his rules. It was a suggestion saying: 'You're watching sports, while your friend, Ted, does the same and your wife is busy.' 'I haven't seen this guy in weeks, in addition he lives not far. What a great opportunity it is!' he thought and dialled quickly the number. 'Hey, Ted! Are you watching the game too? Don't you want to come over, I've got some beer in the fridge.. Great, see you soon!". After putting down the phone, Ed smiled thinking, 'Now I feel close to my friends.'

The above scenario shows the basic idea behind the KRAMER system. Social interaction in telephony can be enriched with a context sharing mechanism (Ed seeing his sister-in-law out of country), on top of which additional personalized functionality can be introduced (Ed's notification rule to pick up his daughter when his wife is unable to do so). But the key intelligence remains to be harvested from the community. By learning situations that many users have defined worth being notified of, the system can derive social intelligence concerning situations that might be important for the community (getting together to watch sports). In consequence, KRAMER may find them pertinent and suggest them to those users that have not defined them by themselves.

The situation of watching sports together with a friend while one's female partner is busy will serve as a reference example in this document. It would also be assumed that the corresponding suggestion comes from three situations previously defined by other users. Table 4-1 lists all those situations, and a couple more. One can see that the resulting notification rule (S) is an effect of some semantic processing of the first three. This process is explained in Section 4.7.

Table 4-1. Reference notification situations from the described scenario

#	action	situation
1	notify me WHEN	I'm watching football, friend is watching football, and wife is at a meeting
2		I'm watching football, friend is watching sports, and girlfriend is busy
3		I'm watching sports, friend is watching sports, and female partner is shopping
4		I'm watching football, and wife is at a meeting
5		I'm reading, TV is on, and wife is at a meeting
S		I'm watching sports, friend is watching sports, and female partner is busy

4.2. Situations model

Contextual situations, being an object of manipulation and suggestion in the KRAMER system, need to be defined in terms of a representation model. This model should be expressive enough to represent complex situations made of several context dimensions related to several entities. As we explain in Section 2.1.2, that kind of representation has not been frequently used in the past research works, leaving situations modelled in most cases as an unstructured composition of several semantic concepts. We, on the other hand, need to maintain the structure of an entity-context relation of respective concepts.

We seek a solution in the area of conceptual structures, one example being Sowa's conceptual graphs [SOW83]. To obtain a structure representing a situation, we start by defining our model with a meta-model inspired by a CONON upper ontology [WAN04]. We distinguish, however, substantials (entities) from moments (context description) as proposed in [COS06]. As a result, we define concepts of a context entity and its context state. Computation entity and person are subclasses of an entity in general. Location and activity are domain specific moments.

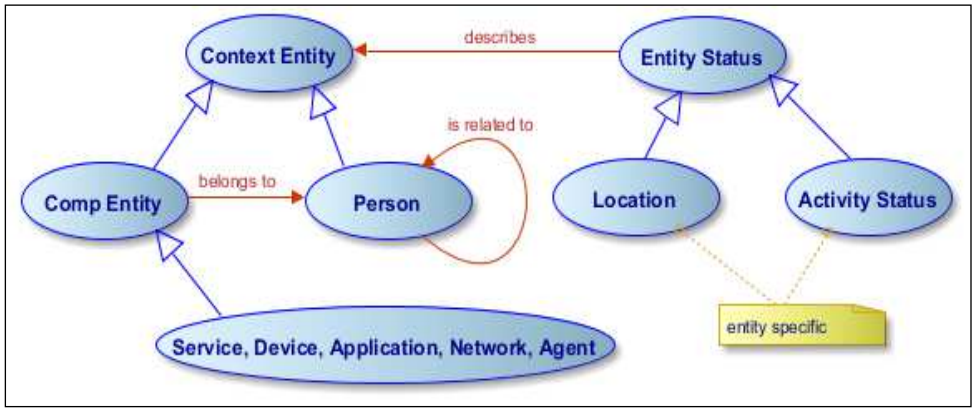

Figure 4-2. The meta-model of situations

We also introduce relations between moments and substantials (*describes*), and between entities and a person (*is related to*, *belongs to*) to model the fact that one's situation is in fact one's context along with context of his or her close ones. As a result, by instantiating concepts representing a situation and by inferring the respective meta-concepts relations, we receive a conceptual graph, a conceptual tree to be accurate, an example of which we present in Fig. 4-3. This conceptual graph has a "me" concept in its root. We say that this complex situation exists with respect to one particular entity, a person who perceives the situation.

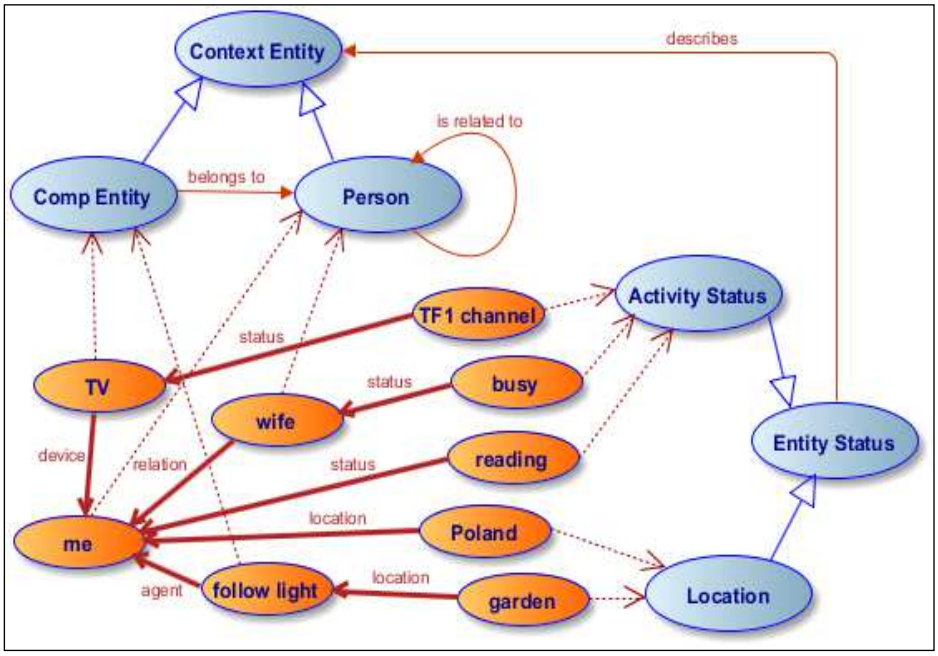

Figure 4-3. Inferring a conceptual tree from the situation meta-model

Following the notation in [CRO07] we define a situation conceptual graph (tree) SCG used in the KRAMER system along with its support. It should be noted that concept types are of four kinds (four concepts in the meta model) and relations are connecting either two entities or an entity with its context (dotted arrows in Fig. 4-3).

Definition 1. A support is a 5-tuple $S = (T_H, \{T_E\}, \{T_L\}, \{T_S\}, T_R)$, where:
- T_H is a finite, p-ordered set of human relations types $(T_H, \leq_H)$;
- $\{T_E\}$ is a set of finite, p-ordered sets of entity types $(T_E, \leq_E)$, *e.g.* services, devices, applications, agents, *etc.*;
- $\{T_L\}$ is a set of finite, p-ordered sets of location types $(T_L, \leq_L)$ specific the entity type;
- $\{T_S\}$ is a set of finite, p-ordered sets of status types $(T_S, \leq_S)$ specific the entity type;
- T_R is a finite set of binary relation types divided into two categories: those connecting entities to other entities $T_R^e = \{relation, device, service, agent, \ldots\}$, and those connecting entities to statuses $T_R^s = \{status, location\}$.

Definition 2. A situation conceptual graph is a 3-tuple $SCG = [S, G, \lambda]$, where:
- $S = (T_H, \{T_E\}, \{T_L\}, \{T_S\}, T_R)$ is a support;
- $G = (V_C, V_R, E_G, l)$ is an ordered, directed graph having edges $E_G = (c_1, r, c_2)$:
  $\forall e \in E_G\ c_1^e \in T_H \cup \{T_E\},\ r^e \in T_R^e \Leftrightarrow c_2^e \in T_H \cup \{T_E\}, r^e \in T_R^s \Leftrightarrow c_2^e \in \{T_L\} \cup \{T_S\}$
  and meeting a condition: $\forall c \in T_H \cup \{T_E\}\ \exists e \in E_G\colon\ c_1^e = c \wedge c_2^e \in \{T_L\} \cup \{T_S\}$;
- λ is a labeling of the nodes of G with elements from support S:
  $\forall c \in V_C\ \lambda(c) \in T_H \cup \{T_E\} \cup \{T_L\} \cup \{T_S\}$; $\forall r \in V_R\ \lambda(r) \in T_R$.

Every concept in nodes of such conceptual graphs is a semantic concept taken from a respective taxonomy. Taxonomies model different context dimensions: human relations, types of devices, locations, *etc.* For example, Fig. 4-4 presents a situation, for which being located in Poland is more relevant than being in any city in particular. As a result, these semantically labelled graphs become much more expressive than basic

situation taxonomies as presented in [ANA06a]. This will also enable logical manipulation presented further in this dissertation.

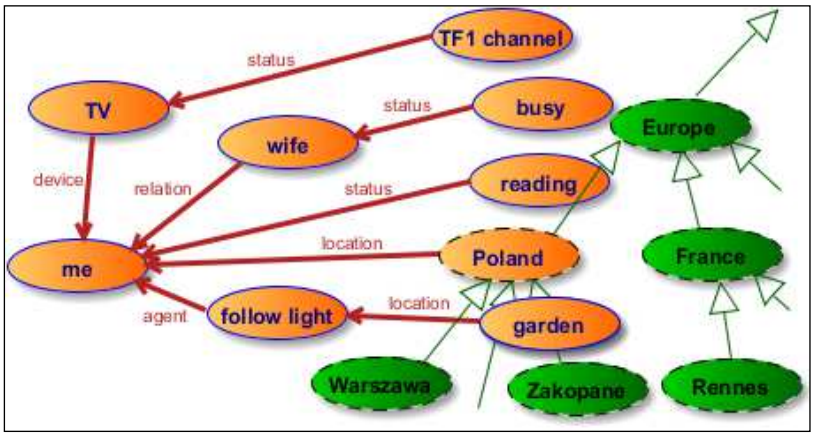

Figure 4-4. Every concept in a modeled situation comes from the respective taxonomy

Moreover, our conceptual graph-based model is consistent with the definition of an abstract situation in situation theory [DEV91]. Indeed, graphs represent only a part of the reality, of the real situation. In fact, every other entity taking marginal part in the situation can be represented as "any" concept, extended further by "any" concept for its context. "Any" is a root concept for every context taxonomy used in our model and is normally omitted in the situation representation.

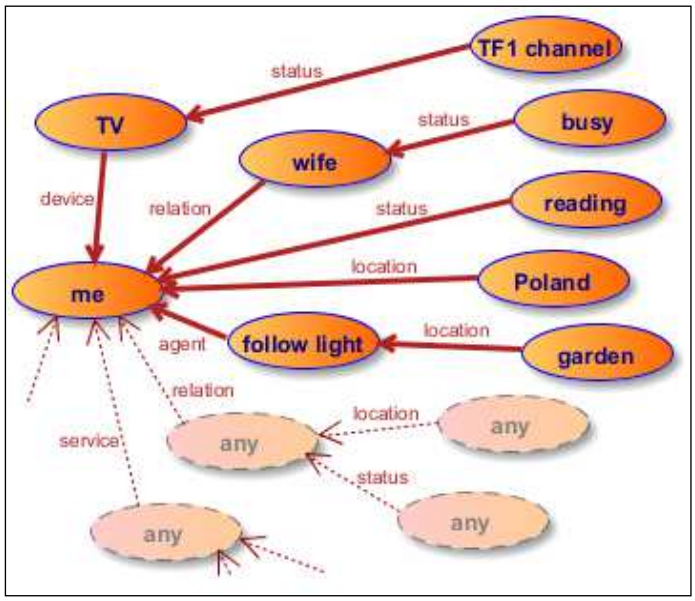

Figure 4-5. A modeled situation is a partial representation of the real one

The motivation for us to select conceptual graphs as a model representing situations was its expressiveness, but also an easy comparison of conceptual graphs, which enables logical operations on situations, *i.e.* their generalization. In fact, in order to reason about situations, understand them, agents need to be able to compare them with one another. They need to measure a degree of similarity between a current situation

and their knowledge about pattern situations. Therefore, a situation model should enable and facilitate this operation.

In case of a plain single concept representation of situations, similarity between two situations is measured as the similarity between two semantic concepts in the corresponding ontology. Comparing two complex situations modelled with a conceptual tree might require measuring similarities of semantic concepts for each context dimension separately, and calculating their weighted mean. Furthermore, it would be a graph matching problem between a couple of situation representations.

Even though optimal algorithms for matching graphs in general are reported to be exponential with respect to the number of nodes in either graph [JIA08], we should remember that abstract situations do not represent the whole knowledge [DEV91]. Instead, the number of nodes is limited to what is necessary for an agent to detect a situation. For instance in [MEI04], "travelling" situation is defined only by using any transportation mean and by the fact of moving significantly. Furthermore, Mugnier reports in [MUG93] that many inter-conceptual graph operations become polynomial, should the involved graphs be restricted to trees. In [ZHO02] the implementation of conceptual trees matching is also reported to be polynomial.

There are also other computationally reasonable implementations of comparing conceptual graphs: one for calculating an ontology similarity based on projection between graphs [CRO07], others presenting comparison of two conceptual graphs as a calculation of their overlapping parts with and without semantic subsumption [MON00, MON01]. We present our algorithm in Section 4.7.2.

There is also another way of looking at the problem of comparing situations. Most of the conceptual graph-based comparison algorithms mentioned previously exploit the fact that concepts in nodes are structured in taxonomies per context dimension. As a result, "chasing an animal" is supposed to be matched with "chasing a mouse" [MON01], rather than "travelling by train", as concept of a "mouse" is a specialization of that of an "animal". A similar idea is to be found in [POO95], where the authors seek for the most interesting common generalization of two graphs in order to evaluate "thematic" similarity between two conceptual graphs.

In fact, according to [MUG93], generalization and specialization are said to be the key computational notions in every reasoning concerning conceptual graphs. Sowa discusses six canonical formation rules as semantic graph-based operators for equivalence (copy, simplify), specialization (join, restrict) and generalization (detach,

unrestrict) of conceptual graphs [SOW08]. These operators can be interpreted by either logical subsumption or graph morphism.

Different researches make use of specialization rules, for instance [LAU07] employs maximal join operator to perform high-level fusion on heterogeneous information represented by conceptual graphs. In our work, we are more interested in detecting and generalizing situations, and therefore finding generalizations of conceptual graphs associated with them.

Mugnier explains in [MUG93] that for one graph to be a specialization of another, there needs to be a projection from the second graph to the first. Projection is a sequence of graph morphisms in a classical graph theory sense, but implying equality of relation types and taxonomic specialization of concept types. As a result, a specialized graph is a super-graph of the original one (external join operation) with possibly semantically narrowed labels (restrict operation), as in the example in Fig. 4-6.

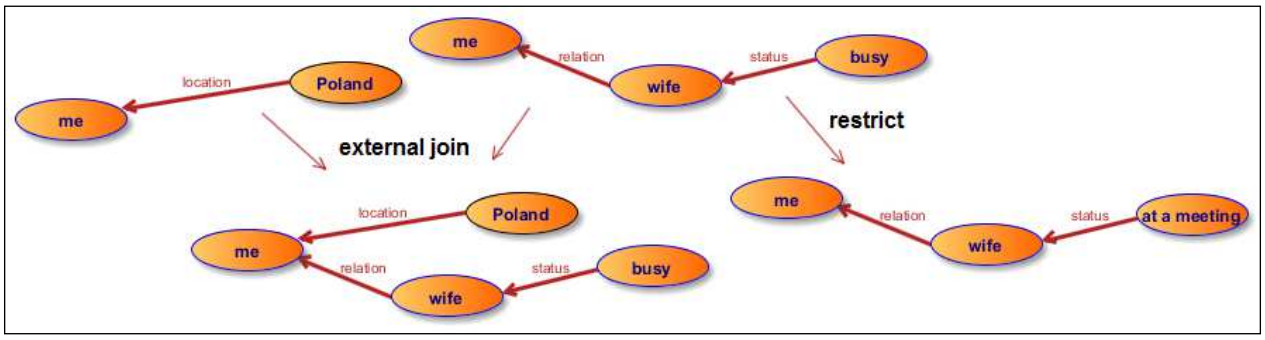

Figure 4-6. Situation specialization with two possible operators shown on our model

This makes the specialization relation a preorder because it is not anti-symmetric as redundant graphs are still possible. Should the injectivity constraint be introduced and internal join operator forbidden, the relation becomes a full order. Therefore, conceptual graphs can form a hierarchy, like in [MUG93] or [LEV92]. As a result, reasoning about relation between two graphs can be transformed into the problem of traversing such a hierarchy. One graph is a generalization of another, if it is an ancestor of that graph.

Considering that conceptual graphs represent situations, reasoning about similarity of two situations is reduced in a way to semantic distance measures as presented in [GAN08] for ontologies. Moreover, finding more abstract / detailed situations implies traversing the hierarchy upwards or downwards. Ye *et al.* introduce this idea in a concept of situation lattices [YE07], [YE08], [YE09]. Although in [YE07] they model situations as simple unitary concepts, similarly to [ANA06a], they notice that this organization reflects the internal structure of situations and is beneficial in identifying situations.

We define therefore a conceptual graph-based situation lattice similar to that of Ye as a semi-lattice of situations with an added bottom element as follows:

Definition 3. A situation lattice SL is defined as $\boldsymbol{SL} = (\{\boldsymbol{SCG}\}, \leq)$, where:

- $\{\boldsymbol{SCG}\}$ is a set of situation conceptual graphs on a common support;
- a specialization relation ≤ is a partial order between situations;
- the top element ⊤ stands for "anything is going on" situation;
- the bottom element ⊥ regroups all the most specific situations.

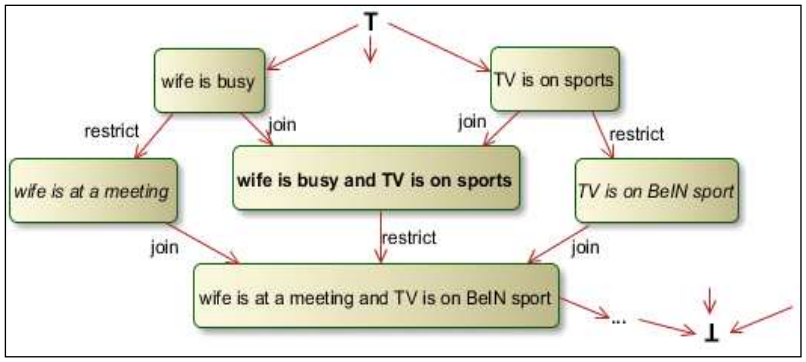

Figure 4-7. A sample part of a situation lattice

The join and meet operators in the theory of partially ordered sets have separate meaning from *join*, *restrict*, *detach* and *unrestrict* operators in conceptual graph theory. In fact obtaining a meet of two situations requires a sequence of restrict and join graph operators on them, whereas the lattice join is obtained via graphs' detachments and unrestrictions. The operators may become a little confusing especially for the name "join", which is shared by the two terminologies. Taking an example of the Fig. 4-7, "wife is busy and TV is on sports channel" and "TV is on BeIN sport channel" situations join in the "TV is on sports channel" situation as an effect of detach and unrestrict operators.

We argue that situations modelled as conceptual graphs can naturally form situation lattices. One such lattice is an abstract object representing all situations possible for an agent to be perceived, taking different context dimensions, with values on different levels of abstraction, into consideration. Situation awareness benefits from seeing a situation space as such an ordered structure. We show its practical advantages in the Subsections 4.4 and 4.7, where technical solutions for situation detection and generalization are inspired by their translation into the lattice traversal problem.

4.3. System architecture

The KRAMER system operates on a client-server architecture. This choice is motivated in Section 3.3. In the case of the telecommunication domain, clients are applications

residing on mobile phones. Specifically, they are contact lists enriched with context. Particular combinations of context dimension values for a user and her close contacts may constitute important situations. Such situations are exchanged with the KRAMER server, which processes them further. In this scenario we also assume that there is another system responsible for exchanging context between contacts in the first place.

For the purpose of the prototype, both systems, the KRAMER and the context distribution one, need setting up a proper infrastructure with a proper architecture. In both cases there is a server unit in the IP network that mobiles exchange data with. Then either the exchange is done only in the IP network or with a use of the GSM network if there is no nearby Wi-Fi access point. The client application isn't in fact dependent on the way the data is transmitted, it is the phone who assures best switching. If no network is available at the moment, the transmission would be reactivated once one of them becomes available.

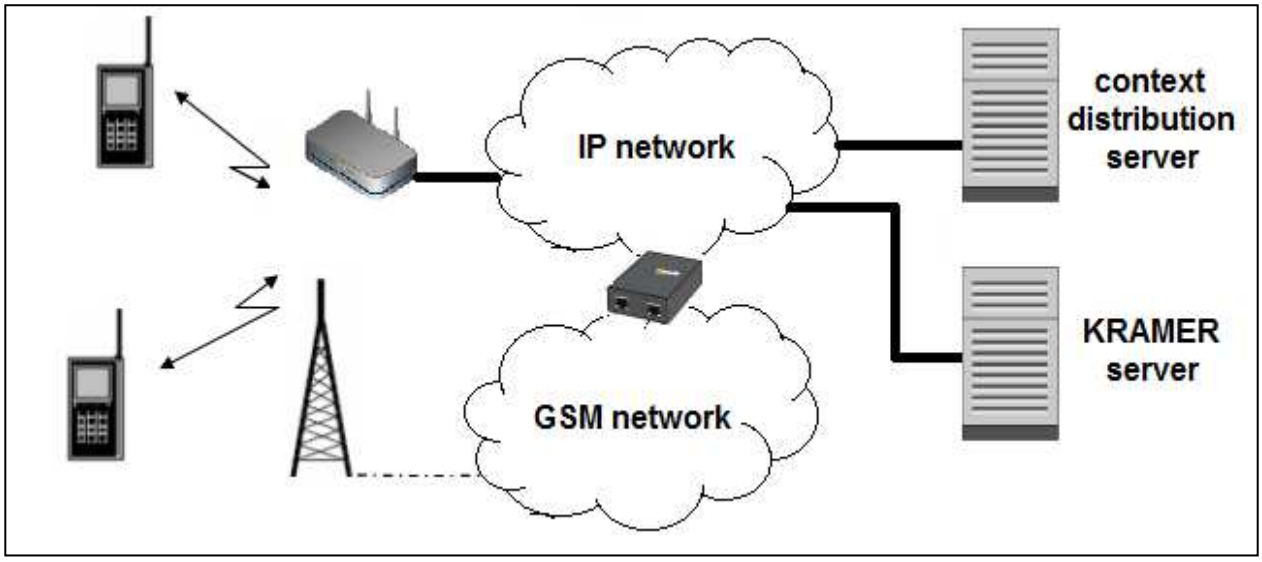

Figure 4-8. The architecture of our prototype, including the context distribution and KRAMER servers

The context sharing server might be aware of the links between users, depending on its implementation. If it had no knowledge of the list of contacts each user has, some additional traffic would be expected. Either way, the information of a user changing context values should be transmitted only to those who are interested. One further limitation could be to send this information only to people one wants to share it with, and with the desired level of details. For those reasons, the data flow would create some virtual links between users that would correspond to the respective graph of contacts, which is illustrated on the social level in Fig. 4-9.

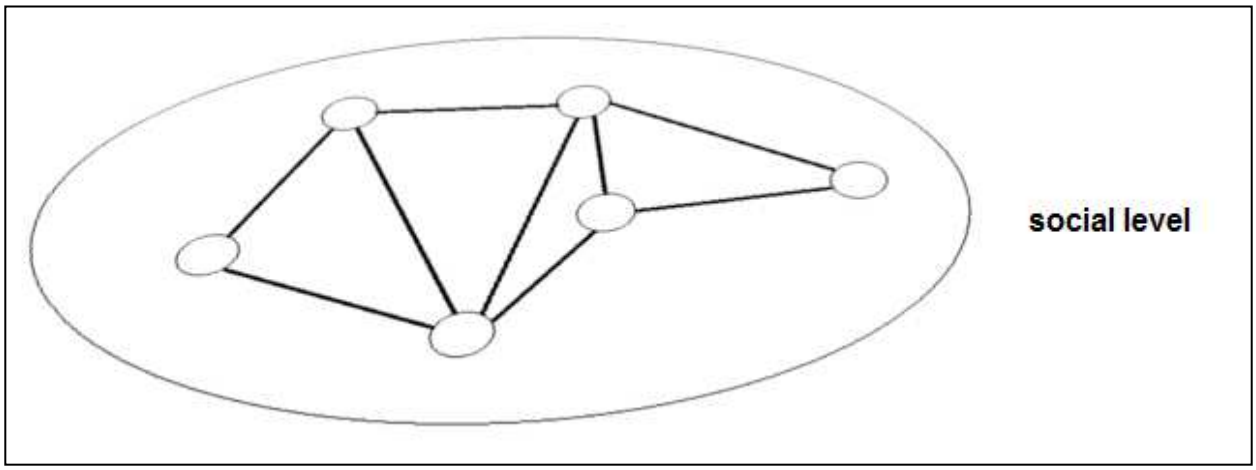

Figure 4-9. Social relations in the context distribution system

However, important situations exchange performed by the KRAMER server would not necessarily result in a similar virtual link graph. In fact, the clients in this architecture have *a priori* no connections whatsoever. They are all in one anonymous collaboration group. However, they would most definitely form those links having no awareness of that fact. The resulting graph would appear somehow chaotic with possibly strengthened links for the "like-minded" users. The latter is however a pure speculation. We have not found any published study on the subject. The fact remains, that any links in Fig. 4-9 would not necessarily match with those in Fig. 4-10.

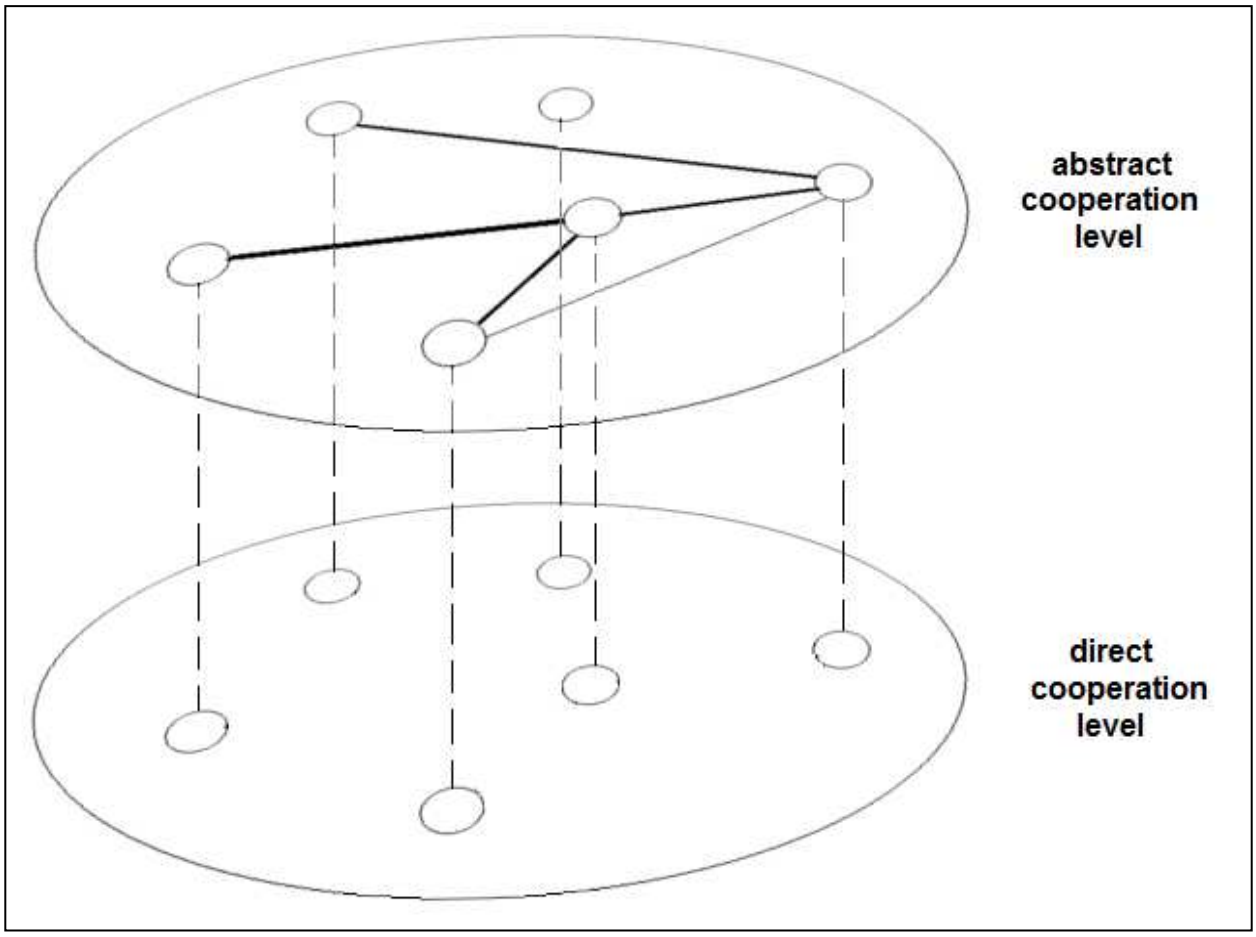

Figure 4-10. Two levels of cooperation in KRAMER, links are formed between like-minded anonymous users

4.4. COSMO client

The software residing on a client device ought to be responsible for both detecting important situations and exchanging data with the system. Following our decision to adapt our prototype to the interpersonal communication domain of application, it should integrate with mobile phone's phonebook application. Any other choice in the future should result in an easy portability of this software, making it an interchangeable module. We called it the COllaborative Situation MOdule, COSMO for short. It is a client of the KRAMER server, which interfaces graphically with a user for her interaction with the system.

The Fig. 4-11 shows the main functional blocks of COSMO in its generic form. The central situation detection element is provided by real-time context data from an outside provider. The situations to be detected have two sources, user's own rules defined via a GUI, or suggestions provided by the KRAMER system. The former are also shared with KRAMER. Finally, once one of the situations is detected, the proper action making use of any necessary smartphone resources, and available at the same time, is performed. COSMO users have further possibility to evaluate the suggestions received by either accepting them (a positive evaluation), or rejecting them (a negative evaluation).

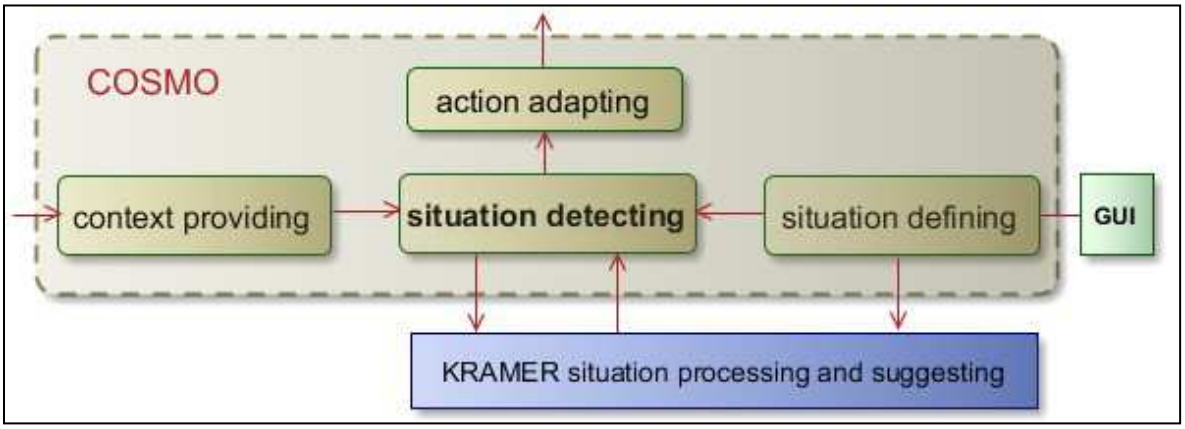

Figure 4-11. The generic COSMO architecture

In the following subsections we discuss the details on the structure and functionalities of our COSMO implementation. The particular nature of context data source is shown in Section 4.4.1. The way the situations are defined and exchanged is presented in mid subsection. Finally, in Section 4.4.3, we explain the technology used to quickly detect situations in function of constantly changing context.

4.4.1. Context sharing

The origin of context data is restricted by neither the COSMO module, nor the KRAMER system. COSMO receives at its input a flow of constant context information updates regarding the source. Those updates should just be structured in triples (entity, context dimension, context value). This way the situation detection element in COSMO will be able to compare the information with parts of situations to be detected. The triple would match or not a branch of a tree modelling a situation. For the details of situation detection mechanism see Section 4.4.3.

Having our prototype applied to the domain of communication between mobile phone users, we narrowed artificially the nature of the contextual data. The data source itself is in consequence adjusted accordingly. We consider that important situations frequently involve other people, especially members of families or close friends. Their context should be equally taken into consideration while making decisions. We propose using a contextual phonebook application and a context distribution system. The former would provide context information regarding each phonebook contact, by using the latter system.

It is assumed by us that modern smartphones are able to sense several context dimensions, and that there exists a logic of initial pre-processing those contextual data in order to obtain a couple of their meaningful semantic representations, which is required as an entry point of our system. We simply state that this is not a part of our research objective. From the point of view of our system, there are no restrictions on the way those representations are obtained, it may be a sophisticated reasoning mechanism (like translating phone profiles against availability statuses, or matching GPS coordinates in places types in Google Places API[6]) [RAE05],[EAG09] or even a user's manual input. For the needs of the prototype we use Google Places API and manual input for mapping respectively location and availability statuses into semantic concepts.

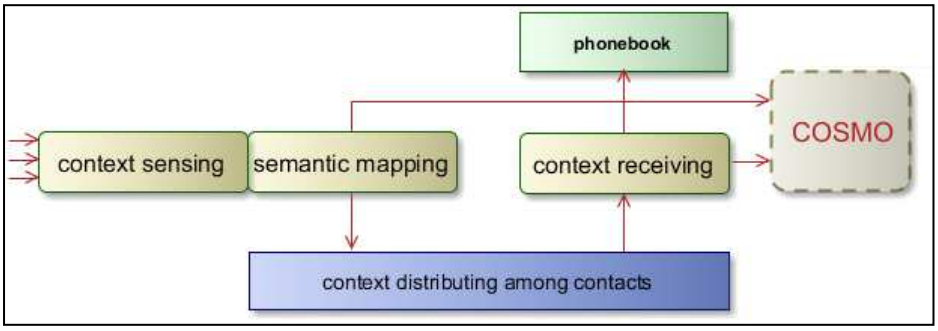

Figure 4-12. Structure of the context distribution system and its relation with the COSMO module

[6] https://developers.google.com/places/

We define the context source for the COSMO module as follows. Sensors embedded in a smartphone capture the readings. Those sensors may be of physical or virtual type [BAL07] and therefore acquire information either about the surroundings of the phone, or about the state of services and applications on the mobile itself. All of those are processed by the logic necessary to obtain two semantic concepts. They can then be shared with the distribution system, the role of which is to inform about updates on other contacts context changes. The latter is provided to both user interface and the COSMO. The process is visualized in Fig. 4-12.

As a result, every user, or her mobile phone to be precise, is able to share her actual location and availability with others via a context distribution system. Every day life in social networks and messaging tools show that people are eager to share a lot of private data in an act of social communication. Studies seem to confirm this fact and talk about new fast mode of chatter and sharing news in communities, which becomes more and more popular [JAV07]. It is hard to tell exactly the reasons standing behind but we consider them being related to the natural need of human expressing oneself, and only recently given some large-scale tools for doing so.

Figure 4-13. Microblogging example on Foursquare [source: Insider]

In a mobile phone contextual information about contacts can be communicated directly in the phonebook. Each contact list entry would be annotated with two context dimensions, concepts describing both one's location and availability status, and with a relationship label. Two former values are those previously learnt by that user's smartphone and then distributed by the corresponding system. The latter has no influence on the classical way of interacting with one another, but it enables later defining contact dependent situations in COSMO. This is static data, defined manually once, in contrast to the dynamic nature of location and availability context dimensions.

Such a context-rich phonebook is a tool for showing the latest, always up to date information expressive for every given contact at all times. Barkhuus *et al.* [BAR08] further compare such sharing applications to the very practice of microblogging. However, neither the context-rich phonebook, nor the distribution system is a novel idea. Context-Phonebook [SCH01b], ContextContacts [RAE05], Connecto [BAR08], and Whereabouts Clock [BRO07] are just four of such applications. Some researchers claim that context information sharing with friends is like story telling bringing closer the contacts to one another, frequently reassuring them [BRO07] and enabling coordination. Others that their application inspires social decisions to contact or not a friend in function of his or her location and availability [BAR08].

Figure 4-14. Contextual phonebook example in our demo

However, in all of the mentioned approaches context dimensions are either limited to just a few possible values (home, school, work, on the way or available, busy, in a vibrate mode) [SCH01b][BRO07], or completely unrestricted (names provided by GSM cell identifiers as in [RAE05], free text location naming as in [BAR08]). We need to find the balance in between the two approaches to "snap the semantic context to the grid" [GRU07]. Therefore, we provide users with a set of semantic concepts for each context dimension (*e.g.* cinema, grocery, boutique, *etc.* for location). Those sets of concepts are large but predefined. If needed, they can be modified and enriched as long as the modifications are propagated through the KRAMER system as well.

Another property of the sets of semantic concepts is that they are organized in taxonomies (*e.g.* grocery, boutique being two types of a store). Fig. 4-15 presents three

upper levels of abstraction of a sample availability taxonomy. Profiles "at a meeting" and "do not disturb" imply both that a user is simply busy. Please note, that this thesis does not provide a formal taxonomy for any context dimension, it rather just uses some sample self-defined ones. Nevertheless, the root of every taxonomy is represented by the most abstract "any" concept. The same applies to the other context dimensions, which is shown in Fig. 4-4, and the mapping is performed by the logic external to the COSMO module. The relationship labels like "wife" or "boss" also form a respective taxonomy with a manual selection of each particular value upon each contact creation.

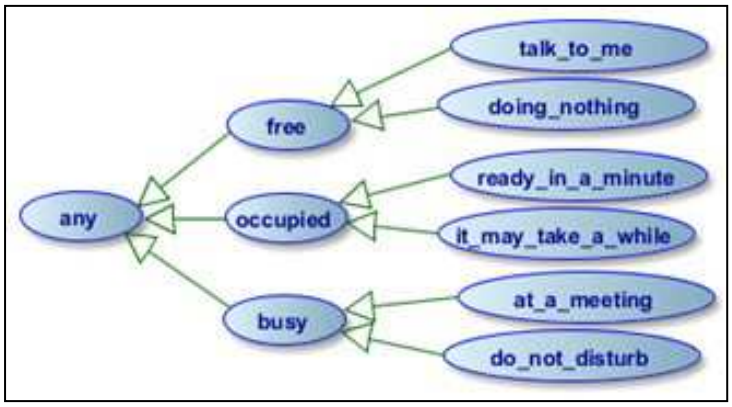

Figure 4-15. Three upper levels of a sample availability status ontology

Once the local context is sensed and mapped to semantic concepts, it is ready to be shared among contacts. Context distribution can be assured by a system using a server as its central element. The server can either keep the knowledge about the relations or ignore it and just follow the addressing instructions of user devices. The first option requires less traffic but stores private data in one spot. The other requires for every context update to be addressed specifically. Probably there could exist yet another, more intelligent way of assuring what is essential - transmitting context data between friends and relatives.

Noticeably, this is a sensitive mechanism as users are disclosing some very private data. Tests performed by some researchers [BRO07] show that people are more reluctant to share their context with location-based services, for example, than with their relatives. It appears that who consumes this kind of data is an important factor. The special nature of close family links motivates Brown *et al.* to limit their service to family members only. Due to the anticipation of privacy issues, the researchers even place their Whereabouts Clock in a kitchen, assumed to be the centre of the family life.

Even though the study of Barkhuus *et al.* [BAR08] shows that initial scepticism regarding disclosure of private context data is not that critical in real application use, we agree that the details that users share should be a function of who can see them

afterwards [LED03]. Therefore, we introduce the possibility of selecting a level of abstractions of the context that one is willing to share with each contact. Phonebook contacts can be divided into groups of the closest ones, which see every detail of a user situation, then those that can see some information, finally those with whom no data is shared. For example, in Fig. 4-14 the first contact shares only the region as a location (Bretagne) without disclosing any specific city or building.

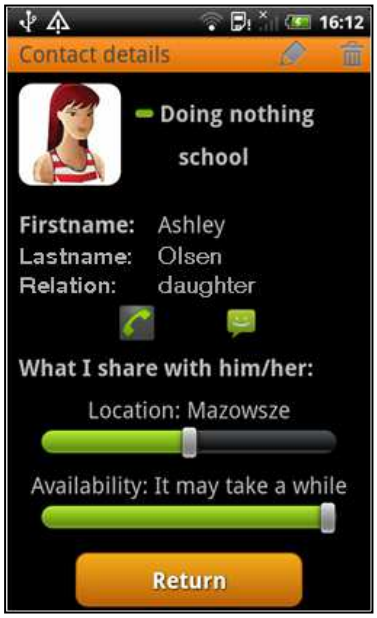

Figure 4-16. Contact details screen in a demo contextual phonebook

As shown in Fig. 4-16, abstraction levels are defined separately for every context dimension of a contact, based on the structure of a corresponding taxonomy, *i.e.* taxonomies can differ by the number of abstraction levels (depth of a tree) and complexity. The preferences regarding the amount of details shared with each contact is likely to depend on the social closeness, friendship or familiar links felt towards that particular individual. Users are invited to create this kind of a context sharing preference table, which should be respected by the distribution system, whatever the technical solution applied there.

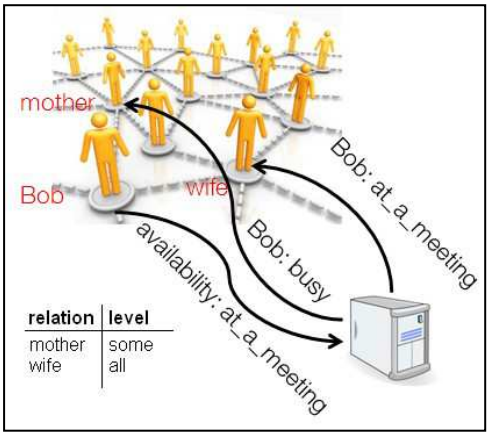

Figure 4-17. Context sharing preference table of a user Bob

4.4.2. Situations exchange

The context sharing described in the previous subsection is already an interesting social interaction mechanism. It enables people to learn at a glance the context of their close ones. Being empowered with associating different pieces of information they are supported in seeing the big picture. In fact, their situation is influenced by those of their friends and relatives. And as a consequence, without much effort they are given real-time data that can make their everyday socially aware and informed decisions easier. A particular context value describing a friend may initiate an interaction, or the opposite, inspire delaying such interaction.

However, a phonebook or other context monitor is not an application that one consults every five minutes. The usage of it has a rather on-demand nature. In consequence, an update on context can pass frequently unnoticed. With some pieces of information missing, the proper association of a situation could turn impossible, and an opportunity to react simply missed. It would be an added value for users it they could define their important situations as a composition of their social context to be notified of each reoccurrence. It would be also a way to structure situations before providing them to the KRAMER system.

The goal of KRAMER is to process collaboratively gathered important situations and suggest notifying those its users that have possibly never encountered them before. One possible way of introducing such situations in the first place would be to adapt some data mining mechanisms to learn them without any user interaction. Such mechanisms are, however, not an object of this thesis. Another, opposite way, would be to ask the users to provide situations themselves. Linking their effort to feed the KRAMER system with their instant benefit seemed like an elegant working solution to employ.

The element described in this subsection is named "situation defining" in Fig. 4-11. A user via a GUI, integrated with the phonebook application in our prototype, may specify, which context dimensions of which entities constitute for an important situation. The situation can be optionally further associated with an action to be performed automatically. For keeping a user only informed for her to decide on an action to take, the default action is a simple notification. However, in its generic form, users define rules of the following structure.

WHEN <situation> THEN <action>

A situation in a rule is modelled as shown in Section 4.2. A list of such rules is stored locally as a reference for the situation detection mechanism. Furthermore, each such rule is contributed to the KRAMER server. The latter is expected to suggest important situations defined by other users, which are also kept as a reference locally upon receiving. Therefore, the situation detection is performed on situations coming from both sources. Keeping a unified form of the rules (i.e. the same set of taxonomies for all context dimensions are stored in every COSMO client and the KRAMER server) enables the interoperability between COSMOs and KRAMER.

In general, rule-based systems, which rely on situation-action rules, can be described as a mean to codify the problem-solving know-how of human experts [HAY85]. They associate a particular production event with a composition of rules. All rules integrated in the system constitute knowledge and a potentially complex if-then reasoning to obtain decisions. These are the so-called ***production systems***, Rete being one example of them [FOR82]. We limit the actions in the prototype to notifications only, which combines the approaches of a mobile phone personalization [KOR04], situation-aware reminder [DEY00b], and context-aware notification service [ETT06].

The power lying behind rule-based systems is responsible for its wide use in expert systems [HAY85], but also in commercial user programmable solutions. On{x}[7] is a recently released application by Microsoft that exposes an API to create simple contextual rules automating an Android mobile phone. Those rules are called recipes and they are inspired by the If This Then That technology (IFTTT)[8]. Motorola SmartActions[9] is yet another example. Those solutions, even though they apply to some quite complex events, are missing a social context to the rules they enable to program.

In our prototype, users are able to define their situation-based rules concerning several contacts from the phonebook list via an integrated GUI. A situation is therefore a composition of possibly multiple local situations of user's contacts. We see the concept of a situation to be social by its nature, and we incorporate it into the COSMO tool. In practice, COSMO users are able to select a number of entities and decide for which context values a given action (notification) should be fired. Furthermore, the values provided may be already taken from upper levels abstractions of corresponding taxonomies.

[7] https://www.onx.ms/
[8] https://ifttt.com/
[9] http://www.motorola.com/us/consumers/SMARTACTIONS™/112638,en_US,pd.html?cgid=apps-software

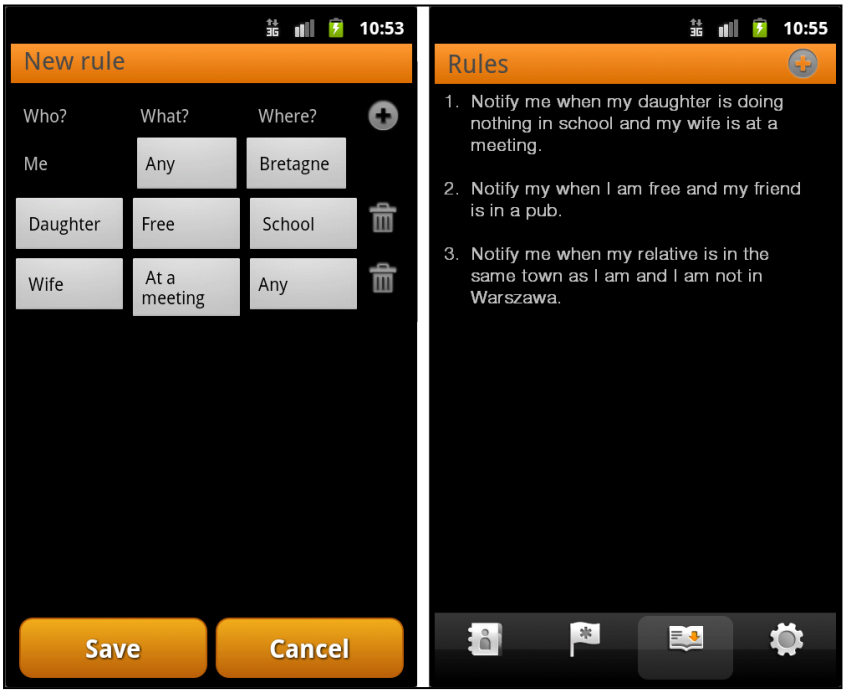

Figure 4-18. Notification rules creation and listing screens in the prototype

Fig. 4-18 presents an example of a created rule in the left and the resulting list of rules in the right. In this example, being in the Bretagne region, rather than in any town or building in particular, is enough for the user to be able to help his wife in driving their daughter home from school. It is important that on the level of rules, contacts are no longer considered as individuals with particular name and phone number. Instead, they are represented as concepts of relation with a user. This way these rules can be shared and reused in the KRAMER system, as explained in the following subsection. But it is for this reason that COSMO requires annotating each contact with a relationship concept.

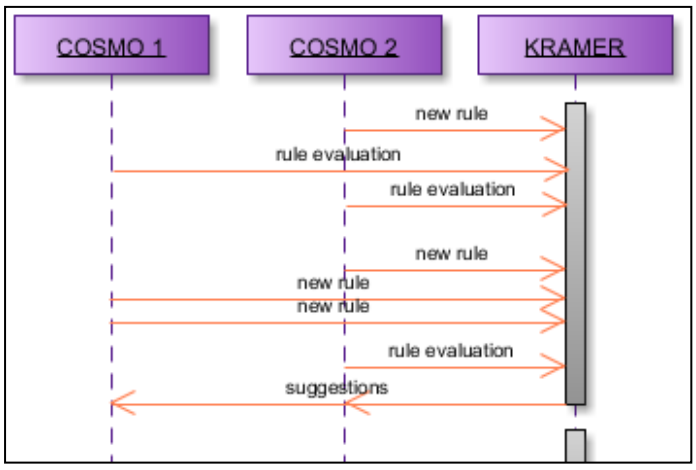

Figure 4-19. Rules exchange between COSMOs and KRAMER

Once a rule is defined it is communicated to the KRAMER server, which in turn shares its knowledge with its clients. Messages with notification rules being defined by a user, those being suggested by the system, and finally user evaluations of the latter

ones are illustrated in Fig. 4-19. The latter type of messages is related to the suggestions received and is discussed in the next subsection. Naturally, since a list of annotated contacts form a profile (of a user as someone who is married if having a contact labelled wife or husband, for example) notification rules relating to a situation not possible to a given user profile may be skipped in transmission.

4.4.3. Action execution

The final task of the COSMO module is its constant monitoring whether the conditions of one important situation, either defined in person or suggested by the KRAMER system, are matched by the current context. In that case a proper action depending on the corresponding rule, for example a notification, should be fired. This functionality is represented in the centre of Fig. 4-11. The "situation detection" element is provided with own rules and those being suggested, as well as with context from the sensors and from the context distribution system. It produces instructions to fire an action associated with whatever situation is being matched, and instructions to cease the action when a context update makes the situation no longer matched.

This approach and the current prototype implementation assume that all actions, even notifications, are of long duration nature (i.e. “keep the notification active” rather than “notify”). Short time execution actions would require defining actions with an apposite meaning, e.g. “increase the radio volume” would be a reaction to “decreasing the radio volume” once the conditions required for the latter action are no more.

The mentioned procedure of triggering appropriate actions requires a mechanism that is responsive and scalable. An action ought to be fired at the very moment the current situation matches one of the reference situations. It may happen that several of the reference situations are co-occurring in one time, therefore the mechanism should be efficient in detecting them all at once. In order to choose a technical solution for that problem, an analysis of the situation detection problem needs to be made. We look into the abstraction of situation lattices in order to learn some practical guidelines.

Following the Def. 3 and the Fig. 4-7 from Section 4.2, all of the possible situations can be made hierarchical in a partial order. In consequence, situations more abstract are closer to the top element of the lattice than those that are more detailed. The specialization of a situation is performed by specializing concepts used in the original one, or by making it more complex, which is represented on the lattice by joining it with another situation. So given that the COSMO module stores all the context dimensions of

all its entities, one might say that its knowledge is placed directly in one of the nodes close to the bottom of the situation lattice.

Assuming that situations defined as important (which users want to be notified of, for instance) are some rather more generalized ones, for example having less than five entities involved, they would be rather located closer to the top of the lattice. This means that a situation defined or suggested can be an abstraction of the situation actually perceived by the COSMO. Therefore, in order for an abstract situation to be detected as occurring at a given time, one of its specializations needs to be perceived. In other words, the relation between a reference situation and the actually modelled by COSMO one is a relation of generalization/specialization.

Furthermore, the problem of detecting that reference situation can be mapped into a problem of traversing a lattice. If a situation lattice was representing the space of all of the situations possible in the world, it would be infinite, for there might be infinite number of entities detailed in the universe. However, given the finite set of entities (contacts in a phonebook) and their possible statuses (concepts taken from respective fixed taxonomies), the lattice itself is also finite. This renders the lattice traversal mechanism possible. One example of a phonebook-based situation lattice is in Fig. 4-20.

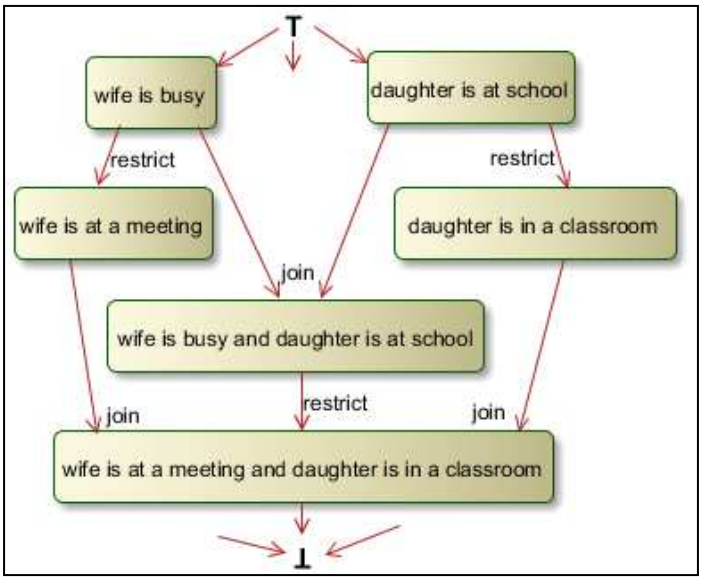

Figure 4-20. A simplified part of a situation lattice from a contextual phonebook situations space

Traversing a situation lattice from the perceived situation all the way to the reference one requires applying a set of situation generalizing operators. As explained in Section 4.2, these would be Sowa's *detach* and *unrestrict* operators. For example, in Fig. 4-20, if the current situation were "wife is at a meeting and daughter is in a classroom" and the reference situation given "wife is busy", one can easily determine that the latter situation is matched and should be detected as occurring at the moment,

for there exists a path of *unrestrict* and *detach* operators (opposite to restrict and join ones), which can be applied in any order. This seems reasonable as whatever the location of the daughter set is, if the wife is at a meeting, she is busy (see Fig. 4-15).

In the implementation of the COSMO module, the current situation is naturally *detached* into particular context triples, one for each context source being updated for a given entity. In consequence, the operators needed to be applied to all those simple situations are first, any necessary unrestrictions generalizing the context conceptual values, should the rule relate to a more abstract semantic value, and second, a join of all necessary context dimensions for the required entities. The composed situation with any necessary context abstraction should be an exact match with the reference situation for the notification, or any other action, to be fired. This stands for getting from "wife is at a meeting" and "daughter is in school" situations in the lattice in Fig. 4-20 to the "wife is busy and daughter is at school" one.

As rule-based systems take frequently the form of production systems in their implementation, we also adapt one of such production systems following the directives obtained from the analysis of situation lattices. The production system Rete [FOR82] associates decisions on performing actions (the productions) with complex set of conditions, situations in our case. The Fig. 4-21 represents a Rete production network composed of two situations: "wife is busy and friend is watching sports channel" and "daughter is in school and wife is busy". The right hand-side part (red nodes) is known as alpha network, and the left hand side (blue nodes) as beta network.

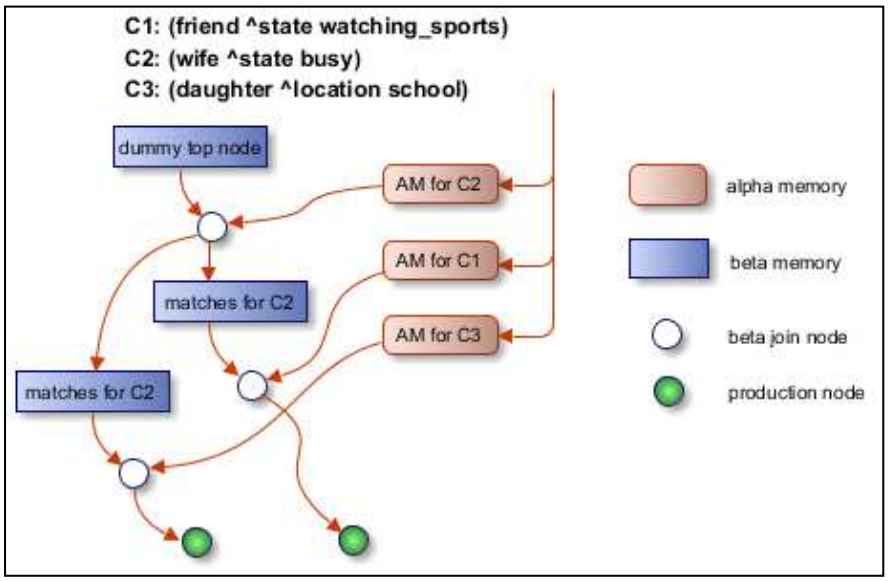

Figure 4-21. A Rete network for two sample situations

In production system terminology, conditions are simple verifiable facts on a property or state of an entity. In case of complex situations, these are the simple ones,

consisting of only one entity and one property, relating to one context source. The situation "wife is busy and friend is watching sports channel" is composed of two such conditions: "wife is busy" and "friend is watching sports channel". They are marked C2 and C1 in Fig. 4-21 accordingly. This corresponds also with the join/detach relation in situation lattices. It is to be noted therefore, that the condition C2 is shared in the two situation, and so is the node in the Rete network.

The alpha network in every Rete network stands for checking validity of each particular condition. The beta network on the other hand is like a logic composition of some conditions. It is like a join operator for simple situations. Whenever several conditions are verified as true, one for an alpha node directly connected to a beta join node and other for all the upper conditions, a new valid composition of conditions is propagated downwards into a beta memory, or the production node. The beta network is therefore performing the join operation, which we defined as necessary of detecting complex situations.

If the system was to detect such combinations only, the classical form of the Rete implementation would be enough. However, we say that context may be introduced to COSMO in a more specialized form than a reference situation would require. If the reference concept is an abstraction of the perceived one, semantic generalization needs to be performed. In order to introduce semantic reasoning as explained above, we enhanced our Rete implementation by replacing the equality (=) condition with subsumption ($\leqslant$) one in alpha network. The condition is said to be true if it is equal or more abstract than the actual data.

Let's take a wife of a user being busy C2: `(wife ^state busy)`. Via a context distribution system associated with COSMO, she shares with him possibly some greater deal of details, for example that she is at a meeting w1: `(wife ^state at_a_meeting)` at work. According to a taxonomy defined, see Fig. 4-15, being at a meeting means automatically being busy. Therefore, even though the COSMO receives information about the meeting taking place at the moment, it is more important for the situation detection mechanism, that the wife is indeed busy. The condition should be nevertheless matched. This is assured by checking for concept descendants in alpha nodes.

The Fig. 4-22 shows a set of COSMO incoming updates about context, which are put into our Rete network. W1 is matched against the condition C2, as being at a meeting is

a more specific concept than being busy. The second join node cannot fire a notification yet, as previous conditions are not matched. The second fact, w2, has no impact on the system, it simply is not matched by neither of the conditions. Finally, w3 is an exact match for condition C1, which is propagated to the awaiting join node. As a result, the respective notification is fired.

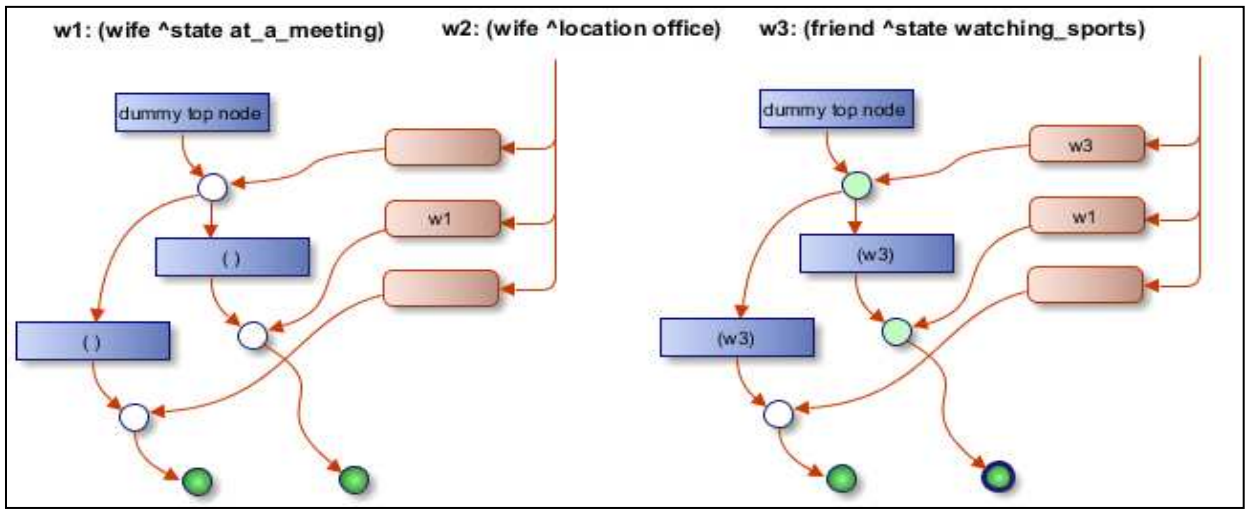

Figure 4-22. Rete network activating with context updates

If at any moment, before receiving the w3 update, the state of a wife changes, into being free for example, then the fact that the condition is no longer matched is propagated through the network. In that case, knowing that daughter enters a mall would not fire a notification. In the same way, having wife or friend restricted their preferences for sharing context with the user, notification can no longer be fired as well. In fact, matching more abstract context concepts against more specific conditions in Rete is simply not valid, *e.g.* knowing that a user's friend is in France does not say anything if he's in front of his TV or not. Whereas his watching a particular sports channel does.

In reality, anytime a user creates a new notification rule in COSMO, or receives one from KRAMER, the Rete network in COSMO is updated. As a result, the networks stored can be more complex that the one in the figure. If there are several situations requiring one particular condition, a respective alpha node is being shared, and linked to possibly several join nodes. The output network's complexity is a function of the number of rules introduced. Nevertheless, the situation detection time is relatively fast. There are at most **n+m** Rete nodes to traverse (n being the total number of situations, and m being the number of conditions in the most complex situation).

Finally, if an action suggested by the KRAMER is the one that is fired, a user can evaluate the pertinence of the recommendation. In case of a simple notification there are

two buttons displayed. One to accept the notification rule, the other to reject it. The decision taken here is sent to the KRAMER server as a feedback for revaluating the situations that it found previously important enough to suggest actions for.

4.5. Communication protocol

The prototype of the KRAMER system is implemented in a client-server architecture, where COSMO modules are clients to one central KRAMER server. The two types of entities in our system exchange messages in order to both gain user collective intelligence and distribute it further among them. There are therefore three interfaces defined for that communication. The first for submitting newly created rules on the server. The second for sharing the suggestions on rules with important situations back with COSMO modules. And finally, the third to provide a feedback on those suggestions, namely their acceptance or rejection.

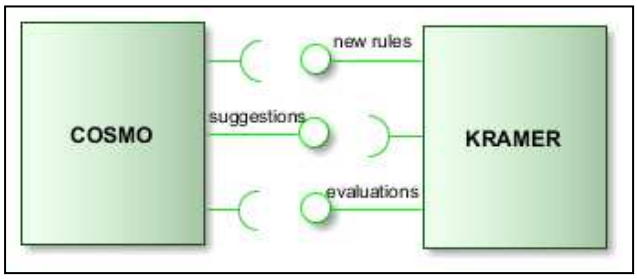

Figure 4-23. Interfaces shared by the KRAMER and a COSMO in their mutual communication

For this reason, there is a communication present between every COSMO module and the KRAMER server. A similar communication is needed for the context distribution system to work, and even though we implement similar mechanisms in both cases, the latter is not a part of the KRAMER system, and shall not be detailed in this dissertation. Nevertheless, the communication in the scope of our system requires a protocol and transmission technology to be implemented between clients and a server.

We have decided to use the MQ Telemetry Transport protocol[10]. MQTT is a lightweight publish/subscribe messaging transport known for its low power usage, which is critical in mobile environments, and is frequently an issue in constant connectivity of social applications. On the other hand, a strict real time message delivery is not a constraint in this case, as exchanging situation-based rules is not a synchronous voice conversation. The KRAMER server is likely to do the calculations on important situations periodically rather than in real-time. Every day use of the system implementing the MQTT proved it to be a good solution. It should be noted that our

[10] http://mqtt.org/

choice for the protocol is an arbitrary one. While there exist many different solutions, for example CoAP[11], which is intended for simple electronic devices, looking for the best one would not have a major impact on the functionalities that the KRAMER system provides.

From the technical point of view, the MQTT message exchange mechanism requires an additional server service. Both the client and a server in the KRAMER architecture subscribe to the MQTT service. They are both treated as clients from this perspective. Then whenever a new message arrives from one COSMO entity, it is forwarded to the KRAMER server, and vice versa. It is a kind of an on demand forwarding of a published message to all subscribers. If there are no messages coming, the service keeps a minimal activity, just listening for the next one to arrive.

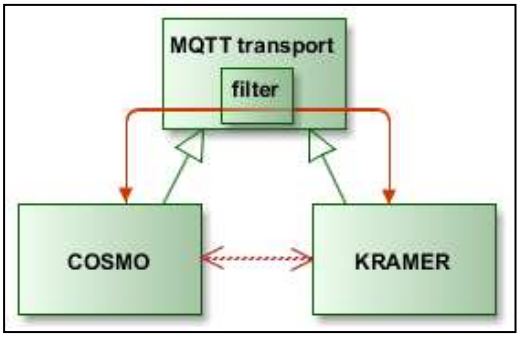

Figure 4-24. A logical communication between the KRAMER and a COSMO

As far as the communication procedure is concerned, upon its first connection, COSMO subscribes with an id (a mobile phone number in our application) on a common channel to a server, which in turn creates an MQTT channel dedicated to that client. The KRAMER server is notified of this fact on its dedicated channel, resulting in its subscription on the newly created channel. From that moment on any message sent by the COSMO module is forwarded to the listening KRAMER. Moreover, any time the KRAMER wishes to publish new suggestion rules, it does so on every channel separately.

In consequence, it may filter out sets of rules for each COSMO from rules involving situations not possible and therefore of less interest for some users. If one has no wife, any rules involving a wife having a particular context would never result in an action. The rule might be very well transmitted and stored on a COSMO locally, as the COSMO action execution mechanism does perform that kind of logical filtering. However, the transmission load could get cut down, should the filtering mechanism be introduced. Because this functionality is independent of the KRAMER server and its

[11] https://datatracker.ietf.org/doc/draft-ietf-core-coap/

collaborative processes themselves, the best place for it would be in the message exchange block, see Fig. 4-24.

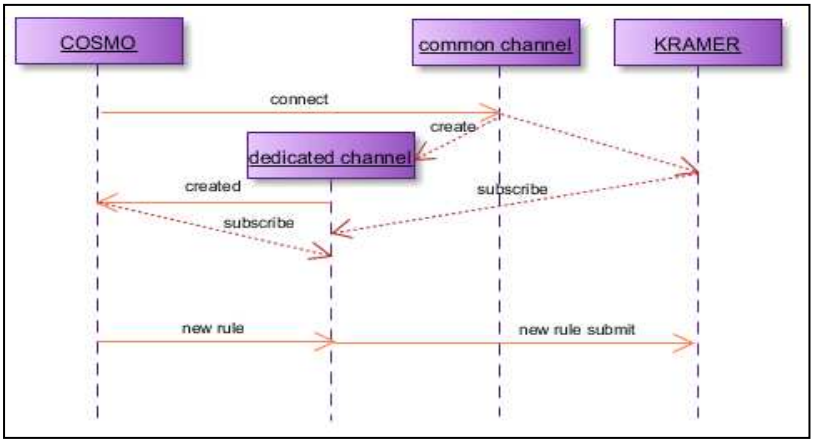

Figure 4-25. A COSMO module connection with the MQTT service

The Fig. 4-25 shows one COSMO connecting to the communication service, and its dedicated MQTT channel being coupled with itself and with the KRAMER entity. The Fig. 4-26 presents a sample message exchange between COSMO modules and the KRAMER server. Those messages are sent as string of characters representing rules with situations, and some additional pieces of information, like evaluation. Naturally, they may be encrypted. For the three types of messages, see Fig. 4-26, not to be confused, each is preceded by a respective key word (*e.g.* NEW, SUG, EVA). The situations themselves are formatted in JavaScript Object Notation[12]. The following is a sample JSON representation of the situation in Tab. 4-1(S).

```
{“person”:”me”,”status”:”watching sports”,”relations”:
[{“person”:”friend”,”status”:”watching sports”},
{“person”:”female partner”,”status”:”busy”}]}
```

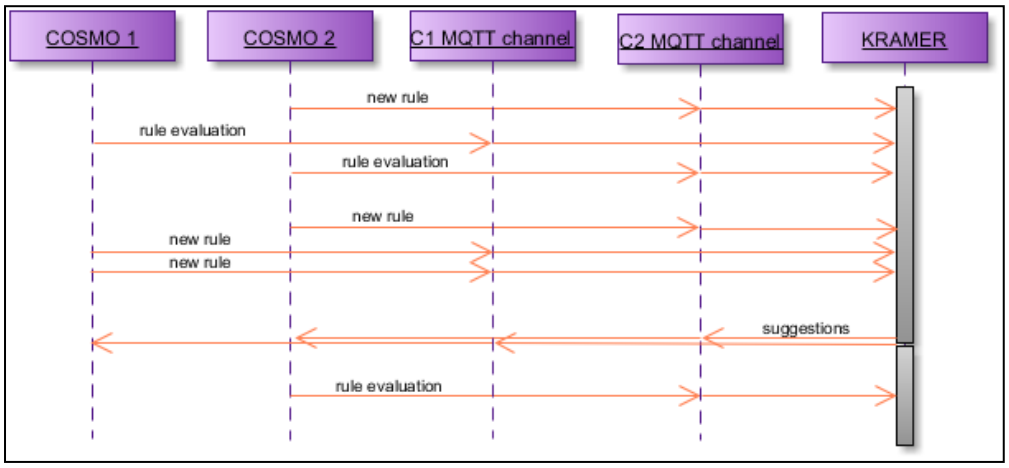

Figure 4-26. Message exchange process between COSMO modules and the KRAMER server

[12] http://www.json.org/

4.6. KRAMER server

The goal of the KRAMER system is to learn importance of situations in a collaborative fashion from its users, and then further share this knowledge with the community, see Fig. 3-1. We argue that even though one might program his or her own action (notification) firing situation-aware service based on past experiences or anticipation of some critical situations possible in the future, one is not very likely to cover all situations that would be found important once occurred. It would be especially difficult as far as situations never previously encountered by a given user are concerned. Therefore, we make the KRAMER system users benefit from the common knowledge of the whole community. The global knowledge of the system is increased with the larger number of rules coming from its many users.

Our architectural choice for the system, discussed in Section 4.3, imposes multiple user clients for one central server, which we call the KRAMER server. The server interfaces with COSMO clients in a threefold manner, as shown in Fig. 4-23. It receives new rules defined by users, it sends back suggestions on important ones, and it permits evaluating those suggestions. Therefore, the KRAMER server is a place, where at first rules with situations that some users find important are sent and stored in. Later they are redistributed among all users.

But situations exchange in the scope of the KRAMER system may be of very different nature, involving different entities, defined on different levels of details. Naturally, aggregating many rules increases the knowledge and skills of the system [HAY85]. On the other hand, users should be able to experience service personalization and not be bothered with every rule that has been introduced to the system. This calls for a logic harvesting those situations, which are important for users in the community. As a result, the collected intelligence will be transformed into a collective one [GRU07], enabling smart recommendations.

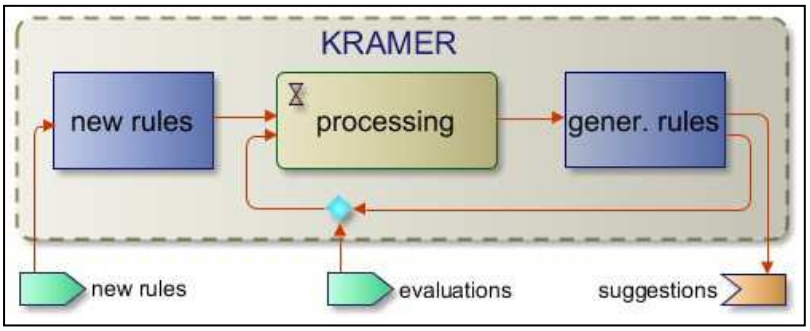

Figure 4-27. The main process of the KRAMER server

The Fig. 4-27 presents a schematic view of the KRAMER server. There are two storages in it, one for the rules contributed by COSMO users, the other for important situation-based rules that KRAMER suggests. The passage from one to another is done via a processing element, which is explained in great details in Section 4.7. The hourglass signifies its periodical execution. There is also a loop going from the processed rules back into the mechanism, for the new suggestions are elaborated based on both the former ones, evaluated or not, and new contributions.

4.7. Important situations

KRAMER is a recommender system for situation-aware rules. Among rules contributed to the server, it needs to find those worth notifying its users of. Therefore, they need to be rules about executing actions in case of important situations. Users are welcome to define rules that apply to the situations they know to be important to them. This does not necessarily apply to situations that one has not encountered before, and may not be aware of their potential importance. For this reason we make KRAMER a system that gathers experience about important situations from all its users. The complete database of rules created by the whole community is stored on the KRAMER server.

Some commercial solutions simply open their rules databases for users to browse through, like IFTTT recipes archives[13]. However, provided that the number of such rules is huge, the problem of users' initial effort to find interesting rules may be too big of a drawback. KRAMER resolves this issue by seeking the collaborative intelligence in processing the stored knowledge, and employing collaborative filtering of that data. Its principle is the same as in other recommender systems widely used in e-commerce [SCH01a], for example. Users are suggested with an interesting piece of information that they were not aware of before, which is harvested from other users' experience.

However, KRAMER rules are more complex beings than the usual subjects of collaborative filtering in its Web applications. Even if rule actions were taken from a closed set of simple actions, the situations themselves are represented by limitless conceptual trees, see Section 4.2. It turns out that the classical implementation of a collaborative filtering mechanism needs to be customized for the needs of the KRAMER system. The process of obtaining both rules to be suggested in the system, and the associated important situations is presented in the following subsections.

[13] https://ifttt.com/recipes

4.7.1. Situation popularity

Items that are recommended in classical collaborative filtering-based engines are in many cases those which have a high rating from like-minded users. The like-mindedness is being calculated by correlation of ratings on other items. A corresponding similarity can be also calculated for pairs of items themselves by means of user-item matrix transformations. As a result, the suggestion is given if the associated item has statistical chances of being given a high note from the recipient, so whether a similar item was already given a high note, or the item is rated highly by a user with similar tastes.

In the case of situations, which are semantic beings, more complex than usual web store items, the question of their similarity is also of a more complex nature. Such situations are not only statistically similar in terms of the pure collaborative filtering, but they may be of different degrees of similarity semantically-wise. This fact has obviously an impact on our suggestion making algorithm. Moreover, we argue some situations to be important and worth notifying of, should they form groups of semantically similar situations that are rated as important in COSMO modules. Naturally, the rating here is binary: having defined the rule or not. So an important situation is a frequently defined one.

Considering that situations contributed in rules on the KRAMER server are modelled as conceptual trees, they are a combination of different context dimension values taking a form of semantic concepts. Furthermore, each concept is taken from the respective taxonomy resulting in different levels of abstractions for the considered information. In consequence, two different situations can have all their concepts the same but one, which is represented on two levels of details. For example, a situation "wife is busy and friend is watching a sports channel" is slightly more abstract than "wife is busy, and friend is watching BeIN sport channel".

The two above situations can be considered very similar. The difference can be frequently of no relevance for situation aware decision making. Moreover, should they be treated simply as different objects injected into the system, both of their popularity ratings would be smaller than their total one. Depending on the granularity of contextual concepts, there may be many more similar situations created by users that make use of different levels of details but mean essentially the same. Despite being

possibly widely used, it would appear that they do not interest but a small portion of users each.

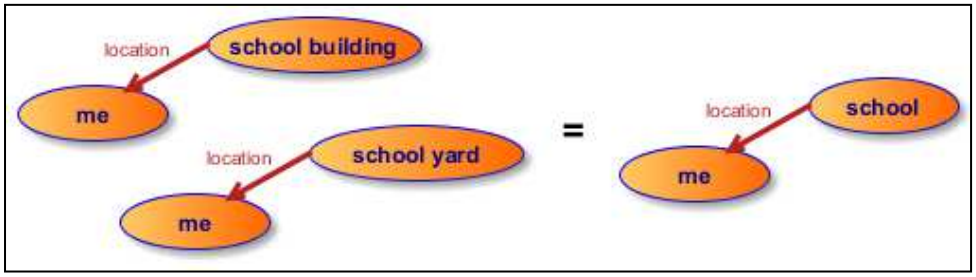

Figure 4-28. Two semantically close situations meaning one generalizing them both

The choice of a very expressive situation model using more or less abstract semantic concepts to express the context has all the advantages discussed in Section 4.2. For the requirements of a recommendation mechanism, however, we need to deal with this sparsity issue. In our algorithm we propose to group situation that are semantically similar. We do not wish to employ any numeric semantic similarity measure, nor an artificial threshold to decide if two situations are still similar or not. We introduce an innovative mechanism, which performs the matching in an opportunistic manner.

In order to consider two situations to be similar, the algorithm looks at all possibly similar situations at once, and evaluates their similarity as a whole. The actual process is explained in the following subsection. If some situations are found mutually similar, the system does not consider but their one representative with a popularity rating being the total of their individual ratings. The situation representing the group is a generalization of all situations implied. The generalized situation is the least abstract possible one, and is calculated by the same grouping algorithm.

The only requirement we put on situations to be similar, is that they need to have the same number of entities and their context dimensions involved. In other words, the structures of the corresponding conceptual trees that model those situations need to match. From the logical point of view, introducing an entity or a context dimension to one situation makes that element to gain importance while it was previously not important at all. The presence of such a factor can completely change the meaning of a situation, which would be further hard to evaluate as the corresponding concept on a former situation would be the most abstract "any" concept.

Finding the generalization of any two situations is always possible, whatever the structure of the corresponding graphs. But a concept representing at once some specific context value and the most abstract "any" value is always the most general of the two, resulting in an uncontrolled abstraction of the meaning. For example, generalizing "wife

is at (location) work" and "wife is (availability status) busy", presented in Fig. 4-29, would result in trivial "wife (is anywhere and doing anything)". This does not make sense with respect to the recommendation system, which ought to suggest meaningful situations to its users.

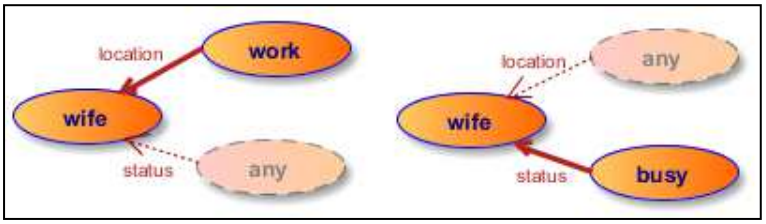

Figure 4-29. Two situations differing in structure

We notice that our choice has some practical advantages. While adding or removing one context description of an entity, possibly eliminating that entity from playing any role in the situation, can result in a situation similar to the original one, implementation logic would become greatly complexified, if not even made impossible. Comparing two graphs or even examining their isomorphism is an NP-complete problem [JIA08]. However, considering only same graph structures reduces in many cases the complexity of the problem to the polynomial dependence [ELL92].

Therefore, the algorithm implemented in the KRAMER is composed of two steps. The first being the situations grouping by matching graph structures with respect to the number of edges and their relation concepts labels. And the second, finding representatives of sub-groups of those groups that are indeed similar by means of their opportunistic generalization. Both of those steps are explained in details in the following subsection, and both of them may be represented as operations of a situation lattice, the idea of which is introduced in Section 4.2.

An exhaustive situation lattice consists of all the possible situations for a given set of entities and taxonomies relative to each of the context dimensions. Given that the "any" concepts are not present in a situation representation, the lattice regroups situations with all sorts of model structures. Making a situation more complex is an effect of a Sowa's join operator, which combines two simpler situations into a more complex one, see Fig. 4-6. The resulting situation is sure to have a structure differing from both former situations. Eliminating all the join relations in the lattice results in creating a family of situation lattices, each dealing with only one specific situation structure. This corresponds to the grouping step of our algorithm.

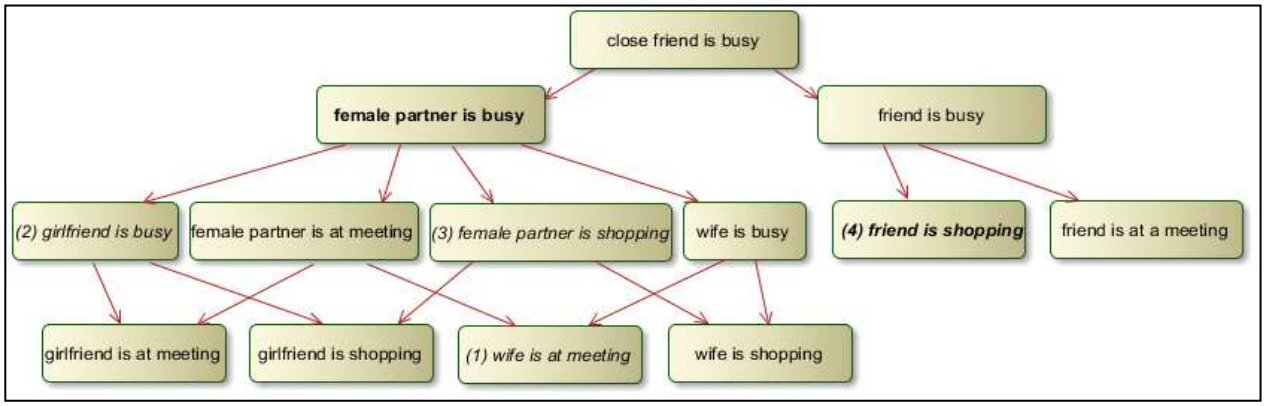

Figure 4-30. A part of a situation lattice referring to Table 4-1

In a scope of one same-structured situations lattice, the relations maintaining the hierarchy are those of restriction or unrestriction of semantic concepts. Finding situations that represent best groups of similar situations in one such lattice is a question of determining the least common ancestors of all situations introduced to the algorithm. Neither of the resulting generalizations, however, can cover a situation that was not introduced. In other words, there cannot exist a case in which at least one path obtained by series of restrictions leading from the abstracted situation to the bottom lattice element does not pass through any of the situations in the initial set or one of their descendants. This, on the other hand, relates to the second step of our algorithm.

For example, Fig. 4-30 presents a case of four situations in a simplified lattice matching a scheme <a person> is in <a status>. Those situations are last parts of those in Table 4-1 with one added, (4)"friend is shopping". By analysing the lattice one may notice that there is one situation, which can represent the (1), (2) and (3). The generalised "female partner is busy" cannot be restricted to any “leaf” node, which is not represented by the initial set of situations. Even if there exists a path from it to “wife is shopping” via “wife is busy”, which is not present in the set, there is situation (3), which covers that leaf. As a result, the abstract situation sums the ratings of the three situations covered. Meanwhile, there are situations missing from the set that prevent including the (4) in the top abstraction, *e.g.* “friend is at a meeting”. Therefore, the situation remains not generalized. As a result, two situations represent those four in the input set, and they are bolded out in the figure.

4.7.2. Generalization algorithm

The situations introduced to the KRAMER system may be more complex than those presented in the example in the previous subsection, see Table 4-1. But the principal idea about their grouping and generalizing similar ones remains unchanged. Moreover,

some practical guidelines can be learnt from the lattice approach. In the algorithm implementation we distinguish also two steps, separating grouping from the generalization itself. They both serve to find similar situations, the first by analysing the corresponding graph structures, the other by adding the semantic properties of the concepts used. The whole process is represented as the centre module in Fig. 4-27.

The first mechanism is based on the adaptation of the algorithm presented in [ZHO02]. The authors there present a formula to measure semantic similarity between two conceptual graphs. It is a recursive formula and for every recursion level it combines all of the possible sub-graphs obtained by cutting a previous level sub-graph's root node. For every such combination it calculates semantic similarity of relation-concept pair for both examined sub-graphs. Finally, the best matching combination of further sub-graphs is selected and passed to the next call level. The result given is a number between 0 and 1, where the latter stands for a complete semantic and structural match.

For grouping purpose, however, KRAMER does not need to calculate the actual similarity value. We simplify the approach and do not bother about the semantic similarity between the concepts in nodes of the compared graphs. As explained in the previous subsection, the grouping step requires finding situations with matching structures only. This can be evaluated for two given situation trees by always having the nodes similarity function $\boldsymbol{sim_n}$ equal "1". With the edges similarity function $\boldsymbol{sim_e}$ taking either "0", if the context dimensions do not match, or "1", if they do, we obtain a function that for every recursion level returns "1", if every edge leading from a current root node has a corresponding edge in the other tree.

$$SoG(n_1, n_2) = \frac{1}{|S|+1} sim_n(n_1, n_2) + \ max\left\{\sum_{s \in S} \frac{1}{|S|+1} sim_e(e_1^s, e_2^s) \cdot SoG\left(n_1^{e_1^s}, n_2^{e_2^s}\right)\right\} \quad (1)$$

We define S to be a set of sub-graphs obtained from eliminating a root node from a current level of recursion (equal for the two graphs), n and e - correspondingly nodes and edges of two compared graphs. Let $\boldsymbol{n_1^{e_1^s}}$ (resp. $\boldsymbol{n_2^{e_2^s}}$) be the root node of the graph obtained by cutting the edge $\boldsymbol{e_1^s}$ (resp. $\boldsymbol{e_2^s}$) from the graph, whose root node is $\boldsymbol{n_1}$ (resp. $\boldsymbol{n_2}$). For every recursion the best sub-graphs' match (the max function of all possible sub-graphs combinations) is selected and normalized to the range <0;1>. If the number of sub-graphs for the two compared graphs is not equal, "0" value is returned. Two graphs are considered as a match if the $\boldsymbol{SoG}$ function of their root nodes has value "1".

While the formula is originally destined for two graphs, we apply it to group all the situations in the KRAMER database. Therefore, it is applied to every new situation introduced into the system. Each situation being grouped has its structure compared by the formula (1) with a single representative of each already existing group, until the matching is found. It is due to the fact that *a priori* all group members have the same graph structures. Once a match is found, the new situation, and the associated rule, joins the group. If, however, all the results are lower than 1 it means a mismatch, and a need to create a new group for that particular situation.

Table 4-2. Pseudocode of the algorithm grouping situations

```
function group_situations(S[1..N])          //S-input set of N situations
   initialize G[]                           //set of groups of situations
   for i:=1:N
      found:=false
      for j:=1:size(G)                      //for every existing group
         if match_graphs(S[i],G[j][1])==1//structures match?
            G[j][size(G[j])+1]:=S[i]        //add a matching situation
            found:=true
            break                           //group found, stop looking
      if found==false
         G[size(G)+1][1]:=S[i]              //create a new group
   return G

function match_graphs(S,Sr)
   if size(S.edges)!=size(Sr.edges)
      return 0                              //different number of edges
   similarity:=1/(size(S.edges)+1)          //default similarity 1 if no more edges
   initialize results[size(S.edges)][size(S.edges)]
   for i:=1:size(S.edges)                   //combine edges types similarity
      for j:=1:size(Sr.edges)               //with a deeper level of recursion
         results[i][j]:=sime(S.edges[i],Sr.edges[j])*
         *match_graphs(S\S.edges[i],Sr\Sr.edges[j])/(size(S.edges)+1)
                                            //formula (1)
   return similarity+sum(results[max(results[])])   //return the best case
```

The pseudocode in Table 4-2 illustrates the grouping process of the algorithm including two functions: `group_situations` and `match_graphs`. The first iterates over

all situations introduced to the system and calls the latter to check the matching structures. If a given new situation matches the first from either of groups it is added to that group. If not, a new group is created. The matching function calculates the best score for each permutation of compared edges (`S.edges`) of two situation graphs. If two edge labels match (`sime(a,b)`), the function gets deeper, until the situation has no more edges. In that case the similarity is said to be always 1 (`1/(size(S.edges)+1)`). If at any point the number of edges is not equal, 0 is returned instead. See [ZHO02] for a running example of this kind of algorithm.

To better show the results of the process, let's take an example of the five first situations from Table 4-1. Even though the situations 1 and 4 have two identical elements, there is one other element from 1, which is missing in 4 ("friend is watching football"). Those two situations will not be in the same group of situations sharing one structure. They cannot be found similar in consequence. Meanwhile, even though the situation 5 seems to be the odd one out, it has the same structure as situations 1-3. It would get probably separated in the second step of situations processing, when the semantic concepts will be considered. As a result, we have two groups, the bigger one regrouping situations of the following structure: `I am in <a state>, <an entity> is in <a status>, <an entity> is in <a status>`.

Table 4-3. Situations from Table 4-1 after the grouping process

#	G	situation
1	I	I'm watching football, friend is watching football, and wife is at a meeting
2		I'm watching football, friend is watching sports, and girlfriend is busy
3		I'm watching sports, friend is watching sports, and female partner is shopping
5		I'm reading, TV is on, and wife is at a meeting
4	II	I'm watching football, and wife is at a meeting

After this first grouping step, the proper generalization occurs. In Fig. 4-30 it is shown that the generalizations derived depend on the actual situations in the initial set of rules. This has been adapted in the implementation. The level of generalization of the result situations is a function of semantic concepts used in the input ones. Naturally, this is performed separately for each group derived in the previous step, but all situations in such groups are processed simultaneously. For our example in the previous paragraph, two executions of the generalization algorithm would be called.

Since the group of one situation only is trivial, the example will follow grouped situations 1, 2, 3, and 5.

The algorithm starts by constructing a graph with the same structure that situations in the group have. This structure is common for all situations in one group. Each graph node is represented here, however, as a taxonomy of concepts of a given type (taxonomy of relationships if the edge leads to the relationship type, taxonomy of locations for the location type, *etc.*), creating a kind of a meta-graph. Then every situation in a group marks its corresponding concepts on this shared taxonomy-based graph. In Fig. 4-31 we present a representation of the example group on slightly simplified taxonomies (e.g. in the upper left taxonomy, which stands for a relation type, situations 1, 2 and 3 marked the same concept, "friend", and situation 5 marked "TV").

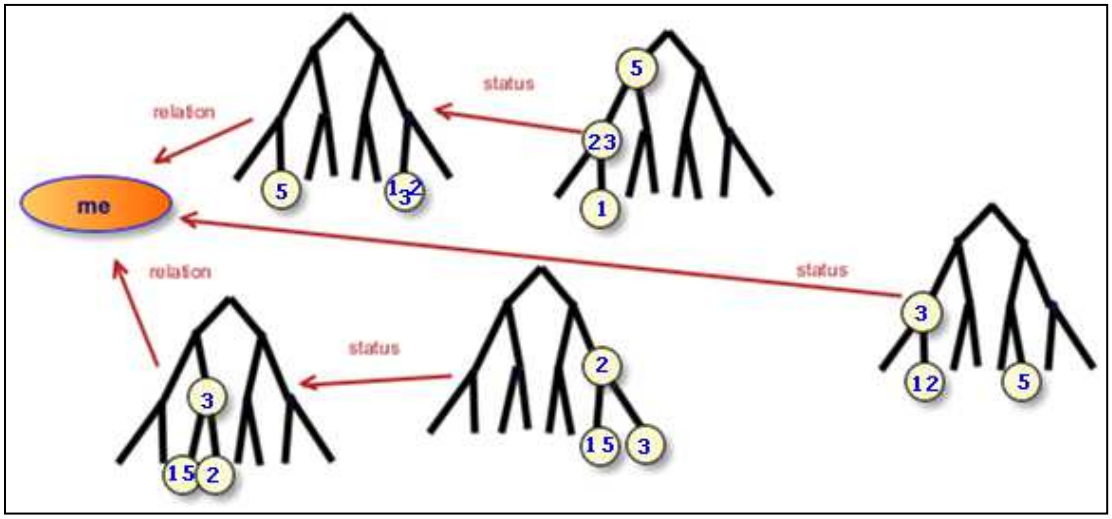

Figure 4-31. A meta-graph structure for the generalization algorithm

One may notice that bubbles with numbers 1, 2 and 3 representing the presence of the corresponding situations in particular taxonomies are always near to each other on those taxonomies. They are related to one another by sibling, parent-child or equality relations. This corresponds to the general impression one might get by comparing situations in Table 4-1, where situation 5 feels to be the odd one out. Our algorithm promotes these close distance relations by finding those situations to be similar, which cover a complete sub-tree with their marks. Only such situations can be generalized into one abstract situation.

To find such complete sub-trees, the algorithm performs successive cutting of those tree branches that point to leafs without marks while having no marked ancestors. Once a branch is decided to be cut of, all of the ancestor branches leading to that one should be cut of as well. This process could be equally done on a situation lattice directly. Our solution, however, uses decomposed structures, taxonomic trees, rather than much

bigger and more memory consuming lattices. Moreover, while the same mechanism is applied simultaneously to every taxonomy-based tree in the meta-graph, it is enough to find one situation differing from another on one tree to say that it is overall different.

The following pseudocode illustrates all steps of the generalization algorithm. They consist of creating a meta-tree, marking concepts of all input situations on appropriate trees, removing node that were not marked and do not have a marked ancestor, and finally finding both maximal uncut concepts and groups of situations contributing.

Table 4-4. Pseudocode of the algorithm generalizing situations

```
function generalize(S[1..N])                    //S-input set of N situations
   initialize metatree[size(S[1])]
   for i:=1:N
      for j:=1:size(S[1].nodes)                 //situations mark values for concepts
         metatree[j].nodes(S[i].nodes[j])[size(metatree[j])+1]:=i
   for i:=1:size(S[1].nodes)
      for j:=1:size(metatree[i].nodes)
         if size(metatree[i].nodes[j])==0 //checking conditions to cut
          && size(metatree[i].nodes[j].children)==0|1
          && no_positive_ancestor(metatree[i].nodes[j])
            metatree[i] \ metatree[i].nodes[j]
   initialize abstractions[size(S[1].nodes)][N]
   for i:=1:size(S[1].nodes)
      for j:=1:N
         if metatree[i].nodes[j]!=null     //concepts remaining get generalized
            abstractions[i][j]:=metatree[i].nodes[j].leastancestor
   initialize groups[]
   initialize used[]
   for i:=1:N
      if used\i==used
         groups[size(groups)+1][1]:=S[i]  //grouping truly similar situations
         used[size(used)]+1:=i            //if all concepts abstracted the same
         for j:=i+1:N                     //for the compared situations
            if abstractions[][i]==abstractions[][j]
               groups[size(groups)][size(size(groups))+1]:=S[j]
               used[size(used)+1]:=j
   return groups, abstractions
```

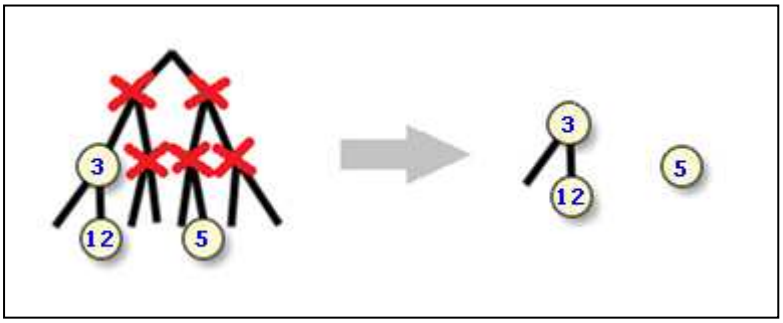

Figure 4-32. Cutting empty branches on one tree example

The Fig. 4-32 presents an family of two trees, one of them being trivial, which is an effect of the cutting algorithm applied to the first availability status tree of the four situations in group I in Table 4-3. Semantic concepts representing watching football (1,2) are taxonomy children of the one representing watching sports in general (3). The concept of reading is further in that respect from the previous three. Therefore, if the tree pruning started eliminating all unrepresented branches, the two groups would get separated in the process. Providing that situations 1, 2 and 3 are found similar on all other trees, they would get generalized, into the "watching sports" concept in that case. Even though situations 1 and 2 relate to football only, situation 3 covers all possible sports concepts, and the three situations generalize as a group. The following table gives the final effect of the algorithm.

Table 4-5. Situations from Table 4-3 after the generalization process

#	**situation**
1,2,3	I'm watching sports, friend is watching sports, and female partner is busy
5	I'm reading, TV is on, and wife is at a meeting
4	I'm watching football, and wife is at a meeting

The presented algorithm will adapt to any set of input situations in its pursuit of the best generalizations. In the worst case scenario, for some completely semantically different situations, it would end up distinguishing every situation, and thus returning the exact same set of situations as it received. Should either of the situations 1 to 3 not be introduced in the example in Fig. 4-30, no abstract situation could express entirely a subset of input situations. Whether it would be a bottom-up or top-down approach of verifying the possible abstractions, the result would be always the same. Similarly, in our implementation objects `metatree[i].nodes[j]` would have at most one element, which would lead to trivial operation of a least common ancestor and groups of situations of size 1. This proves the correctness of our algorithm.

Let's notice that we manage to avoid measuring distances between situations that would require either some arbitrary similarity threshold or much more complicated calculations. Instead, we are able to obtain the situations generalized opportunistically by grouping graphs whose concepts have been found in common uncut sub-trees for all taxonomy trees. In result, concepts remaining in common sub-trees after the cutting process is finished become generalized into the lowest common ancestor. The mechanism works equally well for multidimensional taxonomies. If nodes in taxonomies had possibly more than one parent, like in Fig. 4-33, the generalization would explore multiple paths for finding the least abstract common ancestor. To the extent of our knowledge, this is the first algorithm to generalize large sets of such semantically complex situations.

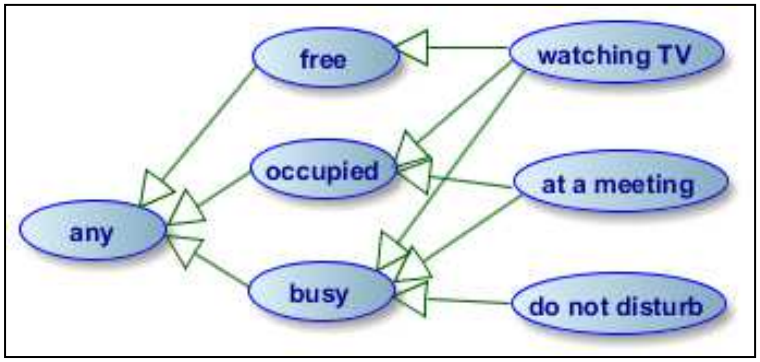

Figure 4-33. A sample multidimensional availability status taxonomy

4.7.3. Rules revaluation

In the example provided by the previous subsection, the KRAMER recommender system would provide a suggestion of the rule "notify me when I'm watching sports, my friend is watching sports, and my female partner is busy". Once the situation happens for any KRAMER user, the COSMO module would fire a notification, as described in Section 4.4. This notification would appear with at least two options for a user to select from. Those options would relate to either accepting or rejecting the suggestion received. In the first case, the rule would be stored locally in COSMO and would fire every time the situation reoccurs. Once rejected, however, the rule is discarded. Furthermore, in both cases, a feedback evaluation is sent to the KRAMER server.

The evaluation interface is specified for both COSMO and KRAMER, and is present in Fig. 4-23. While the positive feedback can be treated as yet another contribution of the same rule, the negative one should be treated in a different manner. Stating that a situation, which was previously abstracted by the KRAMER, is not interesting or

important for a user may say one of two things. Either the generalization or all its detailed situations are not good. In both cases, the negative score should be considered for that rule. In consequence, the reduced rating may influence the level of abstraction of the suggested situation, or even the very fact if the rule will be suggested in the future or not.

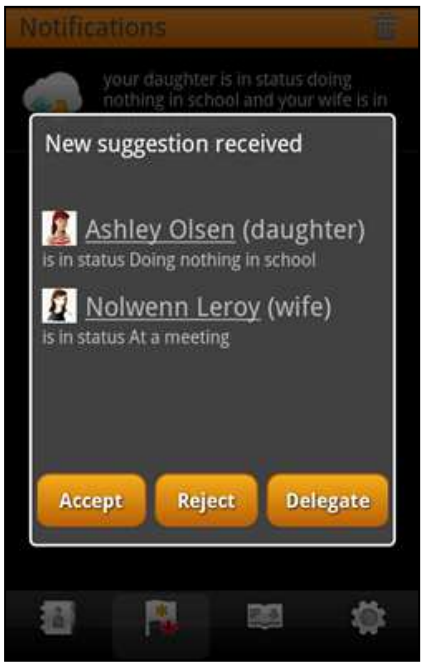

Figure 4-34. A prototype screenshot showing options for a suggestion

Let's consider the Table 4-5 once again, but let's this time consider that one user rejected the suggested rule based on situations 1, 2 and 3. We get a state of the system as in Table 4-6. This time the marking mechanism includes the information about a situation being negative by a minus sign. The process runs in a very same way as before. There may be, however, negative values present, therefore an additional cutting condition is introduced, see the pseudocode in Table 4-8. Fig. 4-35 replaces the actual situation number markings with a sum for both positive and negative representations taken from Table 4-6 and broken into individual concepts. Bubbles with dotted lines contain only the situation 5, which was previously found different from the others.

Table 4-6. Situations from Table 4-3 after receiving a negative feedback

n	**situation**
3	I'm watching sports, friend is watching sports, and female partner is busy
1	I'm reading, TV is on, and wife is at a meeting
1	I'm watching football, and wife is at a meeting
-1	I'm watching sports, friend is watching sports, and female partner is busy

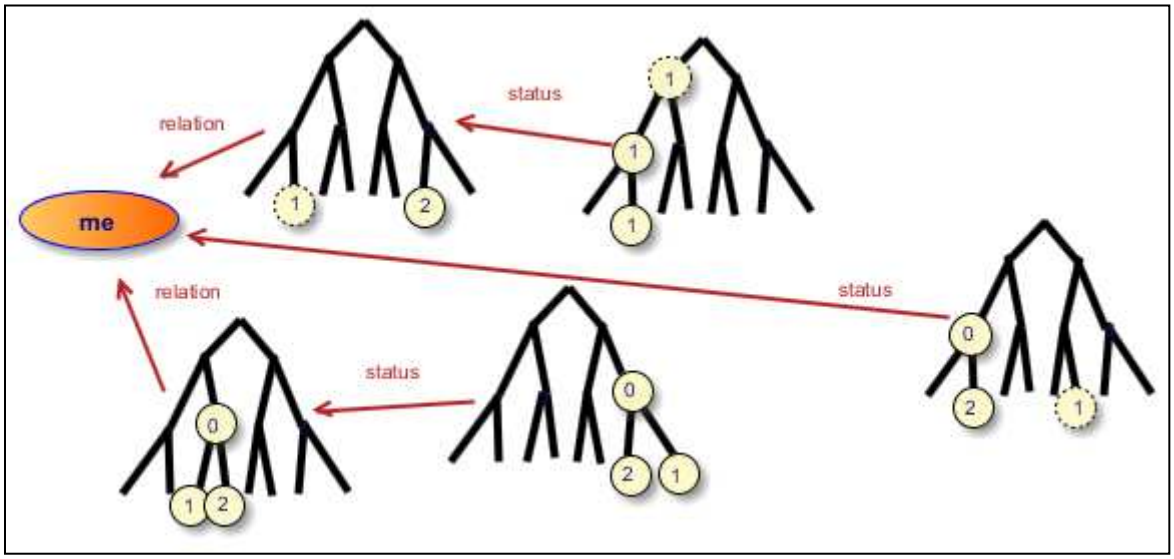

Figure 4-35. A meta-graph structure for the generalization algorithm with negative feedbacks

Judging by the summary values, some generalizations made previously are now considered as unwanted by the community (value 0 in Fig. 4-35). This is especially the case for the second relations of the situations in the example (represented in the bottom of the figure). In consequence those nodes are eliminated from the trees, leaving the pairs of situations 1 and 2, and 1 and 3 separate on the trees after being cut. They would therefore be considered as not similar by our algorithm. Only the situation 2 and 3 would get matched together and generalized. *Nota bene*, it would be the same generalization as in Table 4-5 due to the opportunistic nature of the algorithm. But the corresponding rating is effectively lowered to 2.

Table 4-7. Situations from Table 4-6 after another generalization process

n	#	situation
1	1	I'm watching football, friend is watching football, and wife is at a meeting
2	2,3	I'm watching sports, friend is watching sports, and female partner is busy
1	4	I'm watching football, and wife is at a meeting
-1		I'm watching sports, friend is watching sports, and female partner is busy

As one may notice, the rules have their life in the KRAMER system as they may be revaluated by users, which influences the algorithm for finding similar situations. Therefore, every situation ever committed or evaluated in the system needs to be stored and remembered either separately, or in a complex meta-tree structure. Situations that were once found important may be reduced in importance in time, and *vice versa*. Both positive and negative feedback, as well as a possibility to introduce new rules makes

the important situations set adapt to users and to their changing needs. For this reason, the KRAMER system adapts itself to current trends in the community by its evolving intelligence.

Table 4-8. Pseudocode of the algorithm generalizing situations with negative feedback

```
function generalize(S[1..N])
   initialize metatree[size(S[1])]
   for i:=1:N
      for j:=1:size(S[1].nodes)
         metatree[j].nodes(S[i].nodes[j])[size(metatree[j])+1]:=±i
   for i:=1:size(S[1].nodes)
      for j:=1:size(metatree[i].nodes)
         if (size(metatree[i].nodes[j])==0
          && size(metatree[i].nodes[j].children)==0|1
          && no_positive_ancestor(metatree[i].nodes[j]))
            metatree[i] \ metatree[i].nodes[j]
         if size(metatree[i].nodes[j])>0
          && sum(metatree[i].nodes[j])<=0    //the negative cutting condition
          && no_positive_ancestor(metatree[i].nodes[j]))
          && metatree[i].nodes[j].parent != null
            metatree[i] \ metatree[i].nodes[j].parent
   ...
   return groups, abstractions
```

5. Evaluation tests

In order to evaluate the KRAMER system, *i.e.* whether its definition and implementation respond well to the tasks stated in the first sections of this dissertation, we have performed a couple of different types of tests. As KRAMER is a recommendation system, we have consulted [SCH11] as a reference to decide on the form of the tests to be issued. As a result, they consisted of both off-line and on-line evaluations, including a user study of a relatively small scale. The aim being measuring several properties of the system, as they are presented in [SCH11] and [HER04]. We group them with respect to a four-element scale variating a perspective, from the most user centric on top to the most system centric in the bottom, in Table 5-1.

Table 5-1. Classification of recommender system evaluation properties

↑ user perspective	trust, utility, risk	privacy
	diversity, novelty, serendipity	
system perspective ↓	accuracy, coverage, adaptivity	
	confidence, scalability, robustness	

One classical system property is its accuracy, which is frequently the first recommender systems' comparison criterion. It applies to a measure of to what extent do users agree with the recommendations given. There are multiple formulas adapted to rating items or their ranking. In the KRAMER system the situations are either accepted or rejected, which makes for a binary usage prediction of situations found collaboratively important. The accuracy in this case can be seen as its precision defined as a ratio between a number of true positives (accepted suggestions) and a sum of both true and false positives (all suggestions received).

Whereas accuracy is a measure acquired *post factum*, confidence is rather a degree of trust a system has for its suggestions while giving them. This property can be used by a system to filter out those suggestions that it is not confident enough in, or further research the item. It motivates defining a threshold of whether an item (a situation in our case) is worth notifying of. Trust from a user perspective is her literal belief that a system is worth returning to when performing future tasks. Even if it depends on the accuracy of a system, the parameter, being an effect of a user-system interaction, is of a much subjective nature.

For some systems it may be of further interest to determine the portion of the items covered by its recommendations. Because not only should the suggestions be accurate, but also diverse, novel and serendipitous. Should a system perform well on a whole item space, the suggestions are likely to be found interesting by either of the properties evaluating suggestions in terms of their being different, informative or even surprising. The latter parameters can be measured in a user study, by either enabling an on-line feedback or asking corresponding questions in a follow-up questionnaire.

Two final user perspective properties are utility and risk. In those two the benefits for one using a system are opposed to what can be a negative outcome of either a false positive or a true negative. These two can sometimes even be evaluated without a test by analysing the nature of a system. This applies also to one further system description property. Privacy stands for a degree the user disclosed data are not revealed to other system users, which can be an issue for systems that derive a collective intelligence. This property is transverse to the two perspectives in Table 5-1, as it is equally related to the mechanisms handling the data as to users' perception of a system.

From a more technical point of view, suggestion deriving algorithms should have their scalability and robustness verified. The former is an ability to operate effectively for huge collections of items, which is frequently the case in real recommender systems. The latter is an immunisation to fake data injections into a system, of both malicious and commercial nature. Finally, due to rapid item collection changes and shifts in interest trends, a recommendation system should be adaptive enough to capture those changes, and present its users only the up to date suggestions.

Having presented all the parameters discussed in the literature, only some of them are addressed in our tests. Those are the most important parameters for the KRAMER system, which were also determined by the nature of the tests. The first one, which can be described as an algorithm simulation on synthetic data test, evaluates the system's scalability, coverage, and robustness (Section 5.1). The second one, being a scenario-based user study, covers accuracy, utility, trust, and novelty (Section 5.2). In Section 5.3 we evaluate the KRAMER system by means of results obtained for all the mentioned parameters. We discuss there also an issue of privacy of the data in our system, and the parameters we did not explore in details.

Unfortunately, we were unable so far to perform a large scale user test, which would be very likely to provide even more insight into the usage of the KRAMER system by many people over a long time period.

5.1. Algorithms simulations

Before conducting any user test we decided to perform an off-line simulation of the KRAMER server algorithm performance. The algorithm concerned is the one described in Section 4.7. Its goal is to process semantically a set of situations, grouping and generalizing those that are similar with one another. In other words, it has an input of many semantically rich conceptual trees. As the number of such structures provided can be huge, we wish to examine mainly the scalability of the algorithm proposed. To look into some further properties of our solution, we decided to register also the number of same structured situations groups, and the number of final rules found.

The tested implementation of the algorithm used Java 1.6 programming language. The environment consisted of a two core processor (2 x 2,80GHz) personal computer with 6GB RAM and Windows 7 operating system. Every simulation with a given configuration of parameters, which are described in further paragraphs, was repeated 10 times, the final result being applied with a mean function of those ten. In most cases simulations were performed for a range of 10 to 1000 input situations, giving a clear view of the nature of the scalability function.

Having no existing bases of complex situation data sets, especially using our model of situations, we begin by creating synthetic data. For every simulation a set of randomly generated situations is provided to the algorithm. In order to vary the set, situations are generated with different structures and context concepts, as they were defined by many users with different needs. We limit however the number of entities potentially involved in a generated situation to 3, as we find it of little probability that people would be likely to define situations more complex than that.

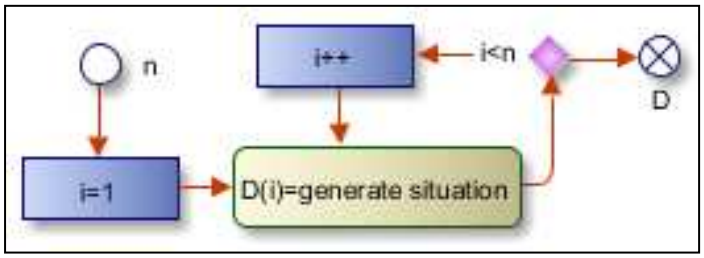

Figure 5-1. Generate random data process diagram

The Fig. 5-1 presents a cycle of generating a wanted number of situations, where the `generate_situation` function is presented in a form of a pseudocode below. There, having by each contact a definite location or availability status is a matter of 50% chance. Furthermore, the number of contacts is also taken randomly from one of the sets R: {0, 1, 2, 3}, {0, 1, 2}, {1, 2, 3} and {2}, respectively for each function in the family,

with an even probability distribution. Lastly, the situation is checked for being empty, in which case it is regenerated.

```
function generate_situation(R)
   empty:=true
   do
      initiate situation
      if rand()>1/2
         situation.location:=random_location()
         empty:=false
      if rand()>1/2
         situation.availability:=random_availability()
         empty:=false
      contacts_number:=rand(R)
      for i:=1:size(R)
         empty:=false
         empty_contact:=true
         do
            initialize contact
            contact.relation:=random_relation()
            if rand()>1/2
               contact.location:=random_location()
               contact_empty:=false
            if rand()>1/2
               contact.availability:=random_availability()
               contact_empty:=false
         while contact_empty
         situation.contact(i):=contact
      if is_empty(situation)
         situation:=generate_situation()
   while empty
   return situation
```

In order to smoothen the mean functions obtained, which in some cases were significantly different from the particular simulated ones, a trimming factor of 30% was applied. As a result, the charts in Fig. 5-2 to 5-4 are averaged functions of 7 simulations

out of 10, for the 3 most extreme ones were removed. Even then the processing time dependence does not seem stable for one of the functions in the family. Nevertheless, all three figures enable us to notice several significant facts.

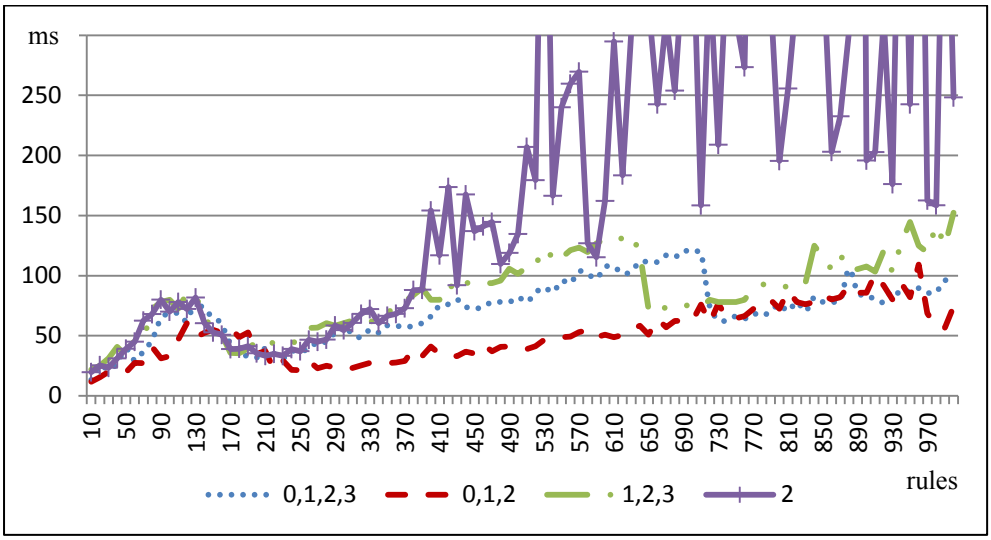

Figure 5-2. Time in [ms] dependency for a number of random situations

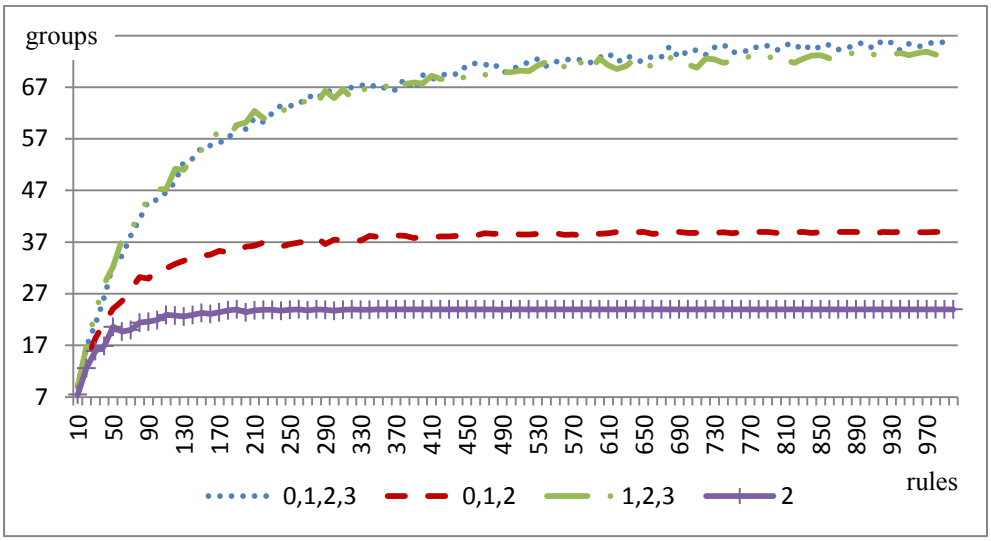

Figure 5-3. Group number dependency for a number of random situations

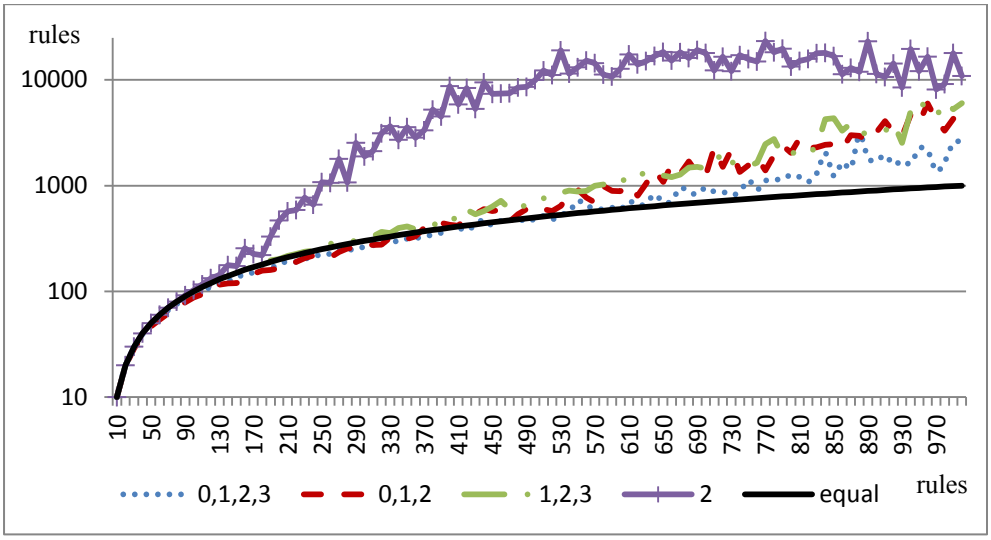

Figure 5-4. Abstracted situations number dependency for a number of random situations

From the time chart (Fig. 5-2) we learn that the actual computational complexity is a function made of several simple functions, which results in stair steps effect. Moreover, the higher the time is, the sooner those steps occur. Also, the bigger condensation of situations concerning bigger number of contacts in situations, the more operations are performed by the algorithm. The extreme example one may see for the situations with 2 contacts only, where the time of processing seems exceptionally random starting from about 300 input rules.

The number of groups found by the algorithm in function of input situations number (Fig. 5-3) is very stable and seems to saturate on particular levels. The less possible combinations of entities involved in situations there are, the faster this function saturates. Those levels are given by the number of all possible situation structures for respective contacts number possibilities. This can be calculated using the recurrent formula (2), where $\boldsymbol{n}$ is the number of graph structures representing all combinations of context dimensions, excluding the relationship context dimension, and $\boldsymbol{k}$ is the number of such structures used (contacts) per situation.

$$\boldsymbol{G_k^n} = \begin{cases} \boldsymbol{1} & \boldsymbol{if\ n = 1} \\ \boldsymbol{n} & \boldsymbol{if\ k = 1} \\ \boldsymbol{G_k^{n-1} + G_{k-1}^n} & \boldsymbol{otherwise} \end{cases} \qquad (2)$$

The number of all possible graph structures for a given set of $\boldsymbol{n}$ and $\boldsymbol{k}$ is in fact a variation with repetition, which excludes being mutual permutations. In that case, the following properties are present. Selecting any number of times a single element ($\boldsymbol{G_k^1}$) gives always only one possibility (aa...a). Selecting once an element from a set of n elements possible ($\boldsymbol{G_1^n}$) is equal $\binom{n}{1} = \boldsymbol{n}$. Selecting k times an element from n elements possible (with repetitions) can be based on selecting k times an element from n-1 elements possible ($\boldsymbol{G_k^{n-1}}$). The possibilities missing there are those involving at least one occurence of the n-th element. Therefore conjunction of one element shorter selection (k-1) of all n elements with that n-th element is added ($\boldsymbol{G_{k-1}^n}$). This justifies the formula (2) and leads to the final group count, where $\boldsymbol{C}$ and $\boldsymbol{R}$ are numbers of context dimensions and relations defined in a situation respectively.

$$\boldsymbol{2^C \left(\sum_{i \in R} G_i^{2^C - 1} + 1 \right) - 1} \qquad (3)$$

The above formula can be explained as follows. $\boldsymbol{2^C}$ is a number of all permutations for a C number of context dimensions to be either describing an entity or not. Therefore, there is always that element present for entity "me". Then, depending on the number of relations R and non-empty selection of context dimensions describing an entity ($\boldsymbol{2^C - 1}$),

the number of combinations is given by the application of formula (2). In consequence the sum of all such combinations for each relation number is calculated. Then there can be one case, where there are no related entities for either configuration of context dimensions for "me" (+1), with exception from one empty, without any context dimensions for any entity, situation (-1).

Finally, the rules number chart (Fig. 5-4) represents a family of functions that is the most unexpected in this case. We present it in a logarithmic scale and with a $f(x)=x$ comparison function. Although there is a fragment for which the amount of rules is actually smaller than the number of initial situations, for the most part it is on a contrary. The number of rules seems to grow exponentially until saturating on a level of all rules possible for a particular number of contacts. The period of such saturation for only 2 contacts function in Fig. 5-4 covers exactly the most random part of the corresponding time dependency in Fig. 5-2.

This is unacceptable with respect to the aim of the generalizing algorithm, which is to provide less but more abstract rules derived from the input situations. Further detailed discussion on this effect, as well as on all other examined dependencies, is given in Section 5.3. For the moment, let's assume that this result is dictated by the random nature of the situations generated, whereas in real life it is not likely that users would declare every possible situation, limited only by the size of taxonomies. Instead, we can assume that the tool would be used in some more relevant types of situations, human "points of interest".

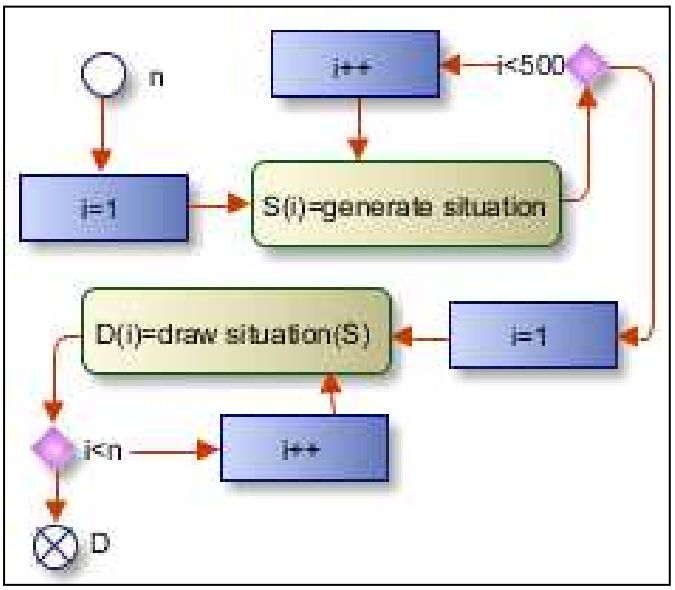

Figure 5-5. Generate focused data process diagram

We explore this hypothesis in a second approach to simulate the algorithm on synthetic data. Therefore, we seek the remedy in limiting the pool of situations that users may find useful to define. In order to generate sets of more focused situations we need to change the generating mechanism a little. Instead of generating every possible situation, we imagine having a set of interesting types of situations and drawing from them. Those points of interest are nothing else than just a fixed set of rules. In result, we add a second step to the data generation process. We arbitrarily set the number of points of interest at 500. The final set of 10-1000 situations is drawn from these 500 initial situations.

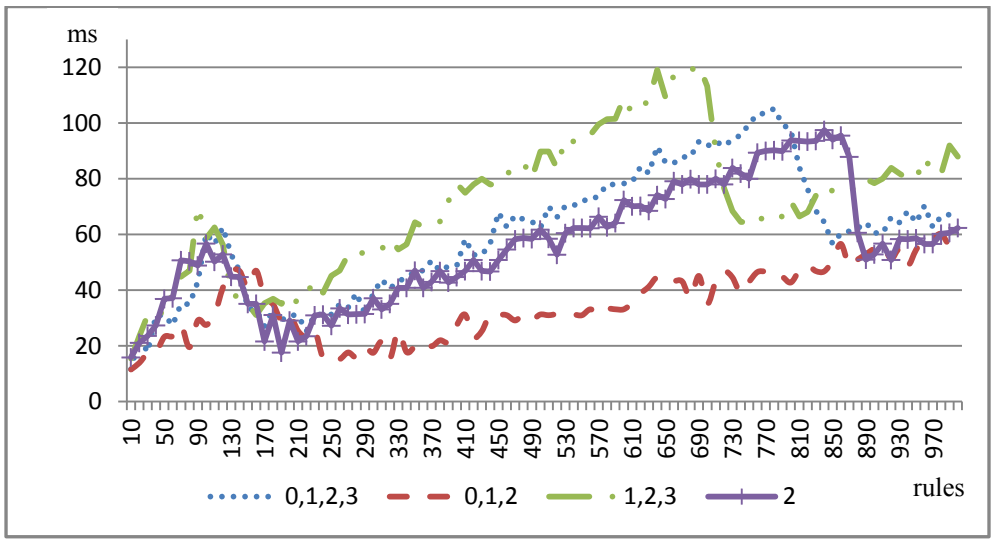

Figure 5-6. Time in [ms] dependency for a number of focused situations

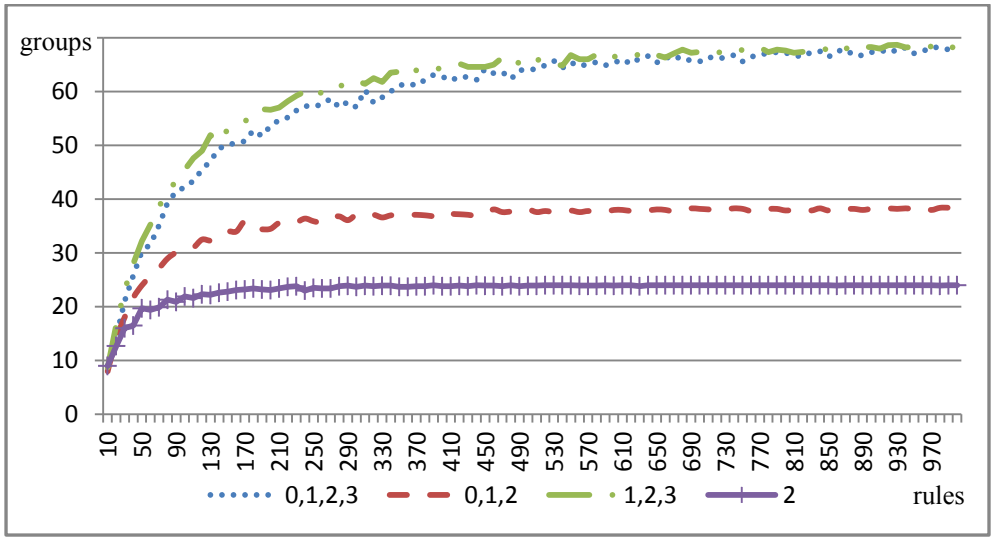

Figure 5-7. Group number dependency for a number of focused situations

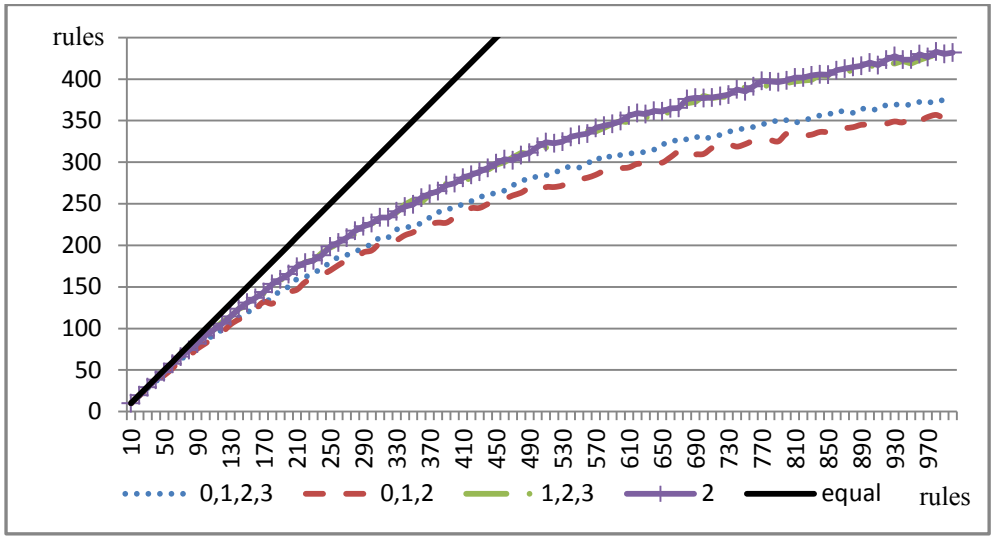

Figure 5-8. Abstracted situations number dependency for a number of focused situations

Looking at the family of plots for the number of rules in function of input situations one can verify that the algorithm works better in terms of its primary aim. Fig. 5-8, presents in linear scale this time a smaller number of rules than the equality function $f(x)=x$. The scenario with limited points of interest, which seems to us to be a more real life-alike case, proves the algorithm to generalize situations into less numerous abstractions. After that, we observe some differences regarding the number of rules that are obtained by different functions for 1000 input situations.

From other charts we learn that the number of groups in Fig. 5-7 is similar in comparison to fully random data, which is dictated only by the structures of situation graphs, and not by the concepts used. But most importantly for the scalability evaluation, Fig. 5-6 shows that the time consumed by the computations seems stable enough. The steps of sudden function changes are clearly noticeable. From the family of plots one can deduce that the computational complexity may depend on the number of situations having more contacts and on the factor of how fast the number of groups gets saturated.

For the rest of the simulations we preserve the focused way of generating synthetic situations. In those situations we examine several parameters that may have further influence on the performance of our algorithm. Knowing that the KRAMER system is used to process context-rich situations that some of its users find important, while others not necessarily, leads us to look into the impact on the algorithm of the following list of parameters:

- amount of points of interest in scale of the whole community,

- ratio of negative decisions on suggestions to all suggestions received,

- share of concepts used in situations that are taxonomy leafs to those more abstract,
- number of context dimensions possible to define in situations,
- distribution of number of entities involved in defined situations.

The default parameters values, if not being tested in a particular simulation, are given in the Table 5-2. As the number of groups being found by the algorithm depends mostly on the number of contacts taken for the generated situations, and since we have this parameter fixed for the following tests, we exclude the corresponding figures and their analyses from the remainder of this section. For each of the listed parameters we manipulate, efficiency (number of rules) and scalability (time) figures are presented and discussed.

Table 5-2. A set of default test parameters

Parameter	**Value**
Number of contacts	{0, 1, 2, 3}
Points of interest	500
Negative situations ratio	0%
Taxonomy leaf concepts ratio	100%
Context dimensions	relation, localization, availability
Distribution of contacts number	25%, 25%, 25%, 25%

Having defined the simulated data generation to be based on a number of pre-generated situations, we first examine the impact of having different number of such points of interest in the community. One may observe an obvious dependence that the more such points are allowed, the more rules can be derived (Fig. 5-9). Still, below the equality function and within limits of the number of generated situation types. There are no big differences as far as the time of computing is concerned (Fig. 5-10).

Simulating a more realistic data set requires assuming that some of the KRAMER suggestions may be found not useful by some users. This would result in having a degree of negative situations in the set to be generalized. The test shows that the algorithm, in case of 20% negative rules ratio, derives fewer rules in a slightly higher time (Fig. 5-11 and 5-12). Should the service offered by the KRAMER system be found useful, we do not envision much more negative decisions circulating in it. Having more negative rules than positive ones would mean that users are not satisfied with KRAMER with little intention to continue using it.

Furthermore, KRAMER users are not limited to selecting semantic concepts only from taxonomy leafs. Defining already abstract situations by users and having the algorithm to operate on previously abstracted rules is a very likely case. The charts show that, for 50% leaf concepts ratio, in a little more time (Fig. 5-13) the algorithm achieves better abstraction efficiency in terms of output rules (Fig. 5-14) than for situations with taxonomy leaf concepts only.

For all previous simulations we assumed having just location and availability context dimensions for each situation contact annotated further with a relationship concept. If we took two more context types, for example time of day (morning, afternoon, *etc.*) and time of year (month, season), users could model rules like: "notify me whenever my husband is in a pub in the morning". In this case the processing time function slope steepness would increase considerably (Fig. 5-15) resulting in some more rules generalized in the process (Fig. 5-16).

Parameters tested in this section were used to model different scenarios for the real test data. Taking into consideration the nature of the data input into the KRAMER system and its generalization algorithm, we assume the parameters experienced in a real test to be close to the values given in Table 5-3. In result we expect the efficiency and scalability functions to match charts given in Fig. 5-17 and 5-18 (in this case points of focus parameter is set to 700, while the range for the number of situations are extended to 7000). Please note that even if the parameter values are not exact, the resulting charts would still give a good estimation on the algorithm behaviour, as long as the real data would display a focused nature rather than a completely random one.

Table 5-3. A set of final test parameters

Parameter	**Value**
Number of contacts	{0, 1, 2, 3}
Points of interest	700
Negative situations ratio	20%
Taxonomy leaf concepts ratio	50%
Context dimensions	relation, localization, availability
Distribution of contacts number	10%, 40%, 40%, 10%

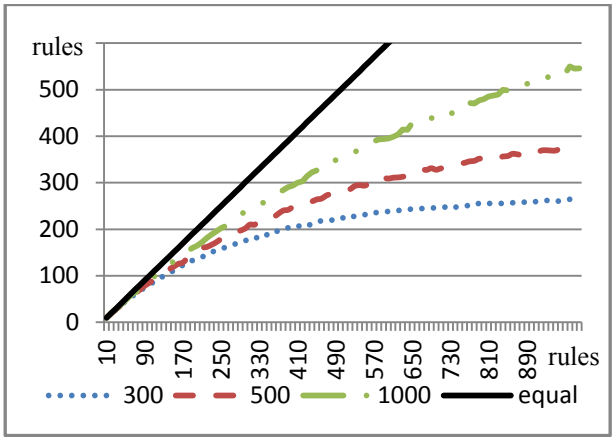

Figure 5-9. Abstracted situations number dependency for different number of focus points

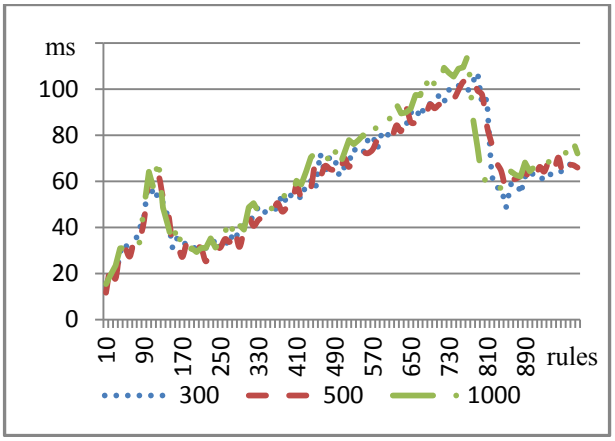

Figure 5-10. Time in [ms] dependency for different number of focus points

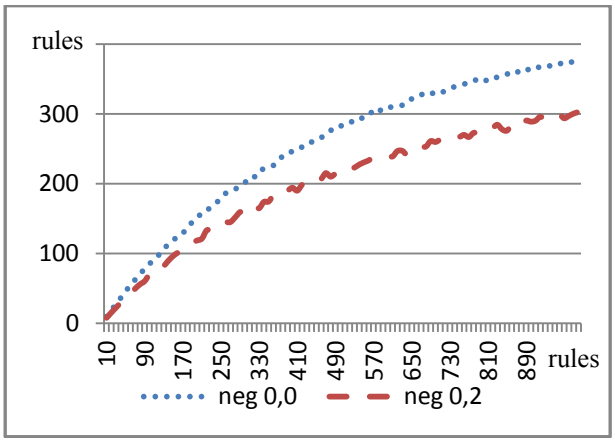

Figure 5-11. Abstracted situations number dependency for different ratios of negative rules

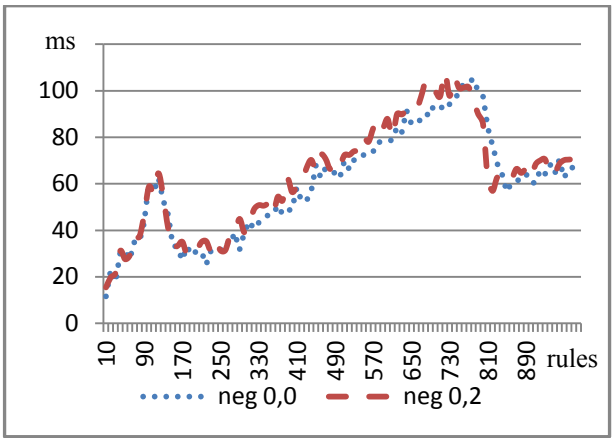

Figure 5-12. Time in [ms] dependency for different ratios of negative rules

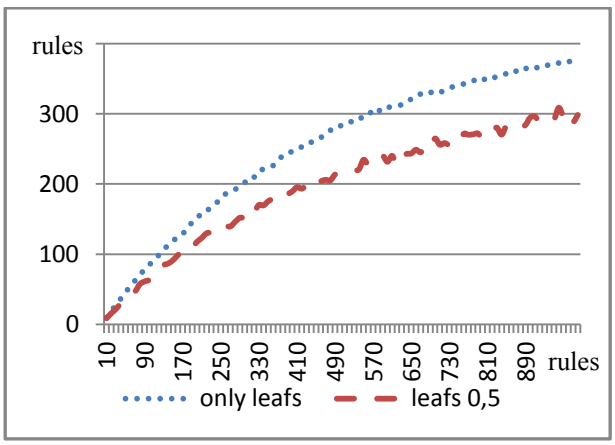

Figure 5-13. Abstracted situations number dependency for different ratios of leaf concepts

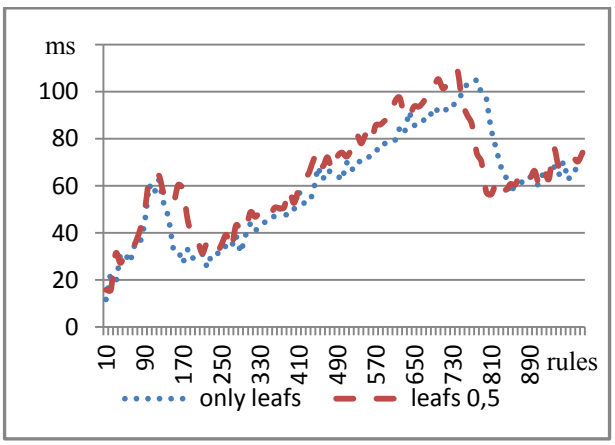

Figure 5-14. Time in [ms] dependency for different ratios of leaf concepts

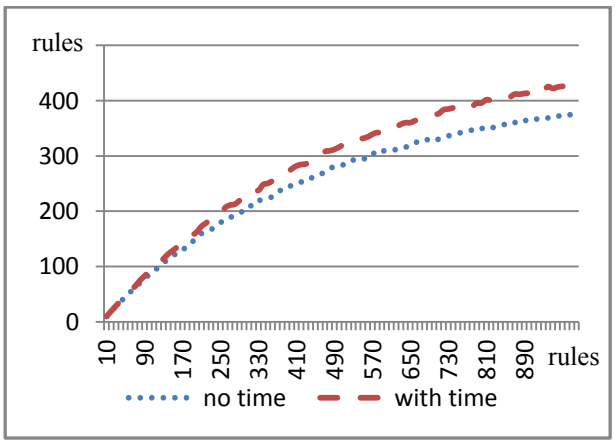

Figure 5-15. Abstracted situations number dependency for different context dimensions

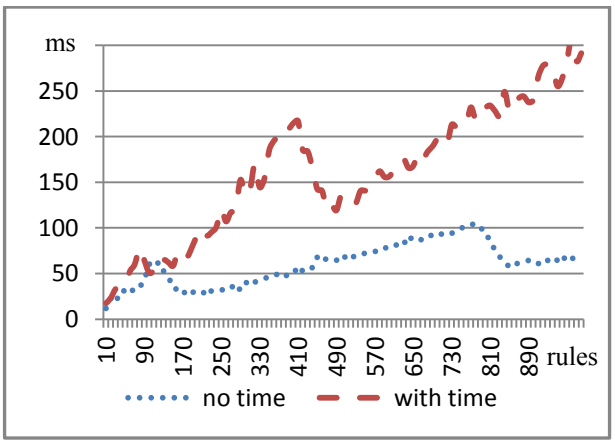

Figure 5-16. Time in [ms] dependency for different context dimensions

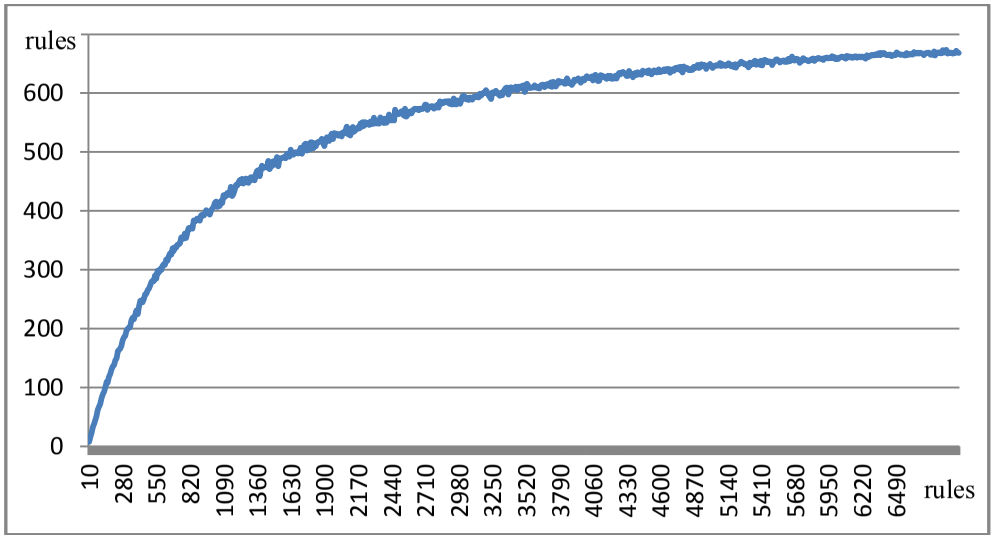

Figure 5-17. Abstracted situations number dependency in the final simulation

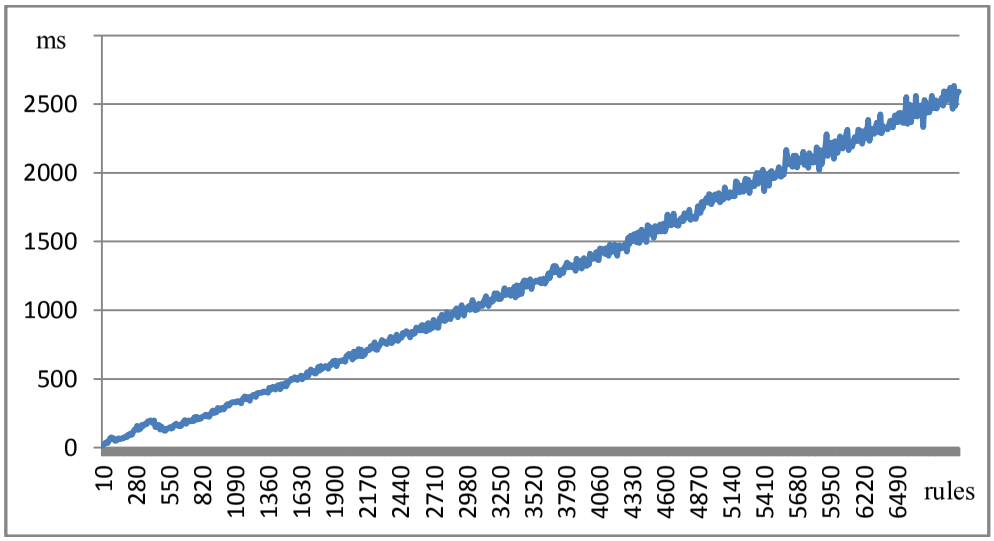

Figure 5-18. Time in [ms] dependency in the final simulation

5.2. Treasure-hunt game

Large-scale tests of systems provide tons of log data for multi-purpose analysis and enable learning system's properties in real use. In the case of a recommender system operating on a completely new type of objects, such as context-rich situations, gathering the necessary data is likely to take long months. Unfortunately, we did not have means to perform such a test. But we did manage to undertake a test of a much smaller nature, which would still provide us with interesting insight. Thanks to a novel scenario-based approach not only were we able to evaluate quickly several aspects of the KRAMER system, but we also acquired experience for preparing better future real user tests.

The main focus of the game-test was the suggestions provided and their appreciation by the users, who received them. Due to the particular nature of object of recommendations, that is users' socially-enabled situations, we needed to study if the KRAMER system is understood, accepted and found useful. Even the very first task might be not trivial as a service, where a well-known phonebook application is turned into a social context exchange tool that notifies of complex situations, can be at least a little confusing. Then, the fact of disclosing private context data to contacts can be also an issue to some.

Therefore, before asking users whether they find KRAMER useful or not, we needed to introduce them gradually to the service. They were invited to learn the COSMO application interface while playing with it, and to discover functionalities of the KRAMER system by simply exchanging contextual information with others, as it is being done in popular social network services. Finally, an effort to interact with the system should provide some concrete benefits to user experience. We packaged all this in a 2-hour test session, which was possible thanks to a preset scenario in a spirit of a treasure-hunt game.

We have invited a couple of groups of testers, 8 people each. Instead of giving them the tool and asking to use it so that we could analyze the logs, we prepared a scenario-based game for them. Our authorial method enabled us to observe most important system properties in a much shorter and more fun manner. The two groups had the same test session programs, both divided into two parts. During the first part, participants learned the interface and functionalities of the COSMO application. In the same time they learned the rules for the game, which was the second part of each test.

The latter was performed in a special building in Orange Labs in Lannion dedicated to testing user experience.

The theme of the game for test users was being a merchant in Brittany. Each one was a merchant of a different good, and they all needed to prepare a reception the following weekend. For a good reception one needs some snacks, some fruits, some sweets, and something to drink. Therefore the merchants travelled between five places in Brittany: Trégastel, Pleumeur-Bodou, Brasparts, Huelgoat and Brocéliande, to find other merchants willing to sell them the missing goods. As there were 8 merchants in total, each of them was selling one type of goods from the four mentioned categories, making for 2 merchants selling goods from one category. To make it more difficult to the players, and more interesting for the system, the regional law permitted selling particular goods only in selected places.

In our game-test, every tester played the role of one merchant. The goal was to find at least one good from three other categories in five days (5 game turns). The theming required adapting the context dimensions used in the system, especially concerning taxonomies related to relationship and location. In the game they became respectively, type of a merchant (Fig. 5-19) and type of terrain (Fig. 5-20). The introductory steps of the test provided the players information as to what did they sell, and where were they permitted to do so. Their schematic sequence is shown in Fig. 5-21, and explanation provided in the following paragraphs.

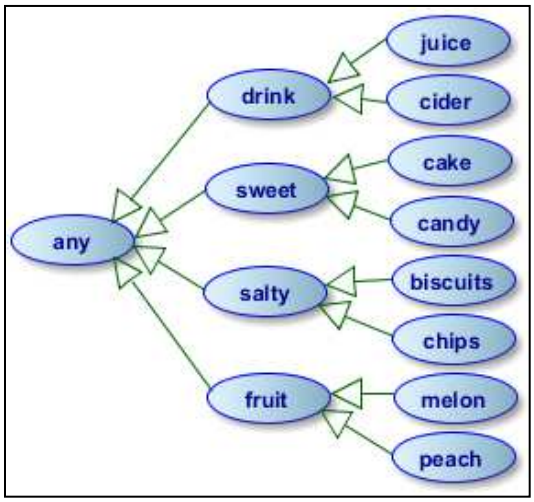

Figure 5-19. Taxonomy of relations in the test game

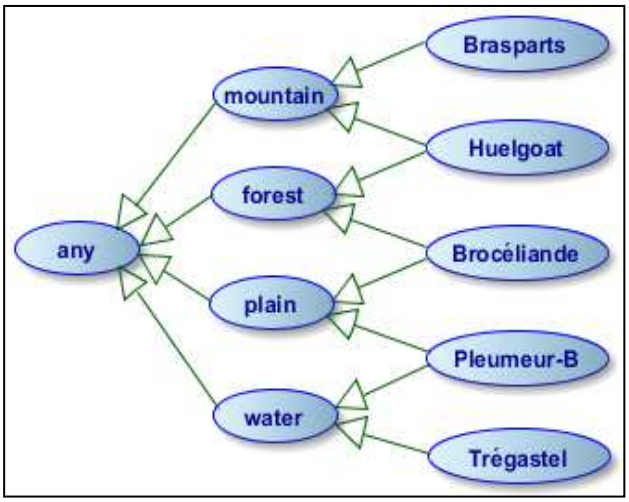

Figure 5-20. Taxonomy of locations in the test game

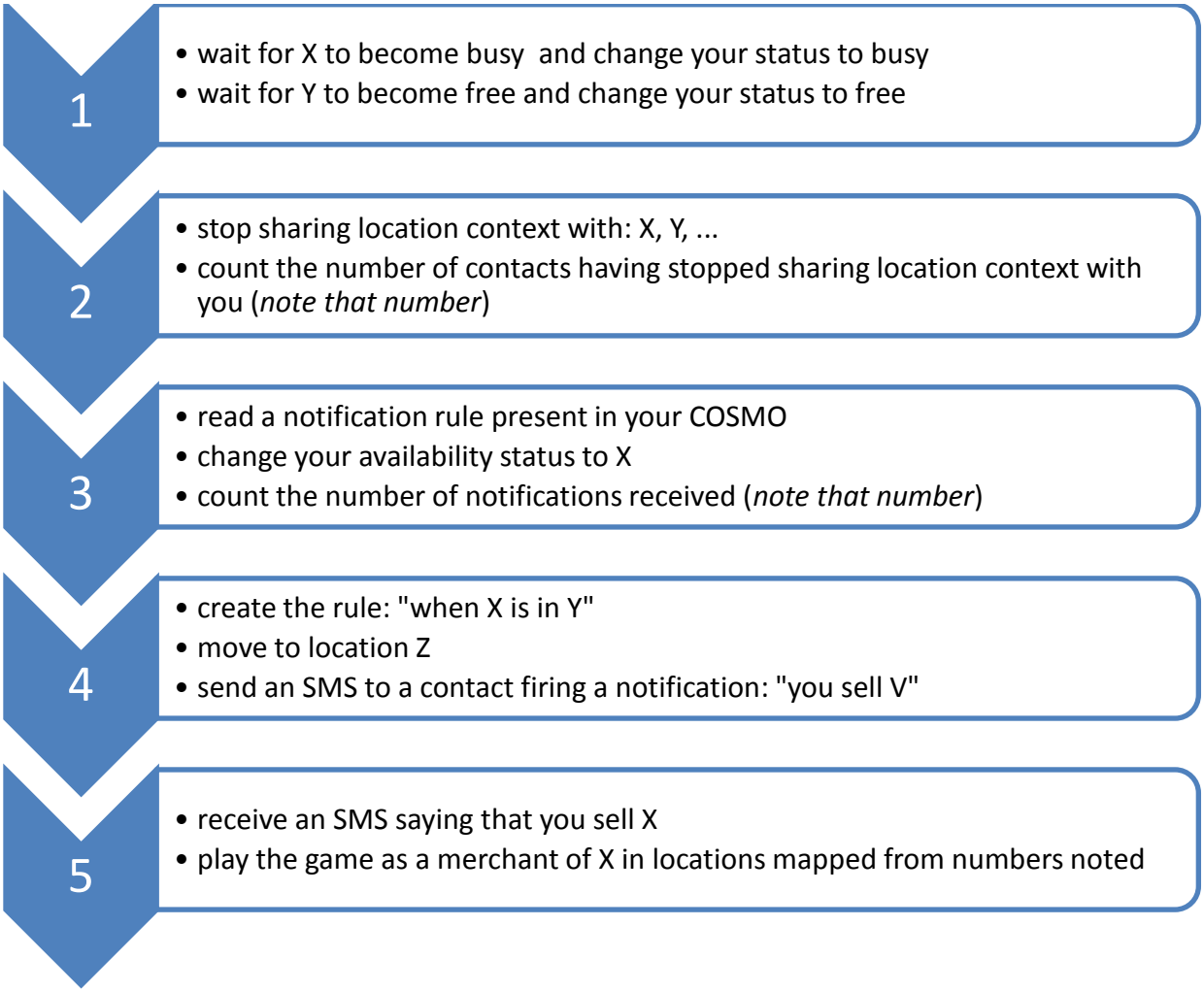

Figure 5-21. The user test steps

Before participating in the game, testers needed to learn the application. However, there was no manual provided to them. The learning process was based on following a list of instructions given to them and not disclosed to the others. Participants were each given a mobile phone with the COSMO application installed, and instructed to interact with the application in a certain way. Even though those interactions were performed locally, they had an effect on experience of others, as all participants were contacts of one another.

By following the instructions correctly, not only did they get familiar with the service, but they were learning the rules for the upcoming game. In the same time we were able to observe and verify, how well did they do with each aspect of the system, and how the latter performed in the course of the game. Every interaction has been monitored during the test and further logged for future off-line studies. Naturally, testers were invited to ask for help whenever needed. This enabled keeping the collective test going, while letting us instantly learn the met difficulties.

The first warm-up instruction was to change the user availability status manually from "free" to "busy" in a particular order. Participants were therefore asked to (1) acknowledge the fact that this particular phonebook enables sharing their status with their contacts, (2) understand that the icons and context description next to their contacts' names signifies current situation of those contacts, and (3) find a way and an application screen to modify their own status. As it was the first contact with the application, and good understanding of the context sharing idea is crucial for the whole service, participants were asked to repeat the task, by "becoming free" again.

As COSMO enables not only sharing context but also deciding how it is shared with different contacts, the second instruction asked to stop sharing location context dimension with certain other users. In other words, testers needed to find a way to change the level of details they would share with some of their contacts into the most abstract “any” concepts. As a result, a certain number of other users would stop sharing their location with them as well. Testers were asked to note that number. Coming back to the initial state required participants to reverse their previous actions once again. From that moment, tester had the knowledge of how to freely manipulate context sharing preferences.

The third instruction involved already the notification rules. There was one already provided in each user's application and was referring to a contact being either ready in a second or ready in a little while, two instances of being occupied (Fig. 7). After having studied it, participants were asked to simultaneously change their status to one of the occupied concepts indicated in their instructions. As a result, several notifications have been fired for all users, one for each contact entering an availability status from the rule. This way, participants have learned that having a rule results in a notification, should the conditions be met. They also obtained a second cipher to note - the number of notifications received.

The final instruction of this test phase required creating a new notification rule. The corresponding situation was given by the instructions and it referred to a type of merchant being in a particular city. Once everyone entered their situations, they were asked to move to a specific room (place in Brittany). As their location context got updated automatically, the proper following of the instructions assured for all of them receiving a notification. Then they were asked to send an SMS message to the person causing a notification with a name of the good he or she was selling in the game. This

way, testers got a feedback on a situation that they entered, which they were able to link to a real action - sending an SMS.

Moreover, they were ready to start playing the game. The two numbers they noted mapped to two cities, in which they were able to sell a good, whose name they received by an SMS. They were only missing one thing - the knowledge that there is a system able to recommend an important situation that they did not define themselves, the KRAMER system. But observing their reaction to that was the aim of the second phase of the test.

The mechanics of the game were simple, each group of three users in line were asked to take their game turn. The turn consists of movement (changing a room) and asking one person in that room two questions of a type: "Can you sell me a cake?", for example. The person asked need to say "yes" if he or she is a merchant of a respective good and if it is a correct place to sell it. Otherwise, the reply would be "no". If a transaction is made, the good represented on a piece of paper is issued. In return, the person asked may then ask back. This way one learns association links between people, goods and places, either positive or negative ones.

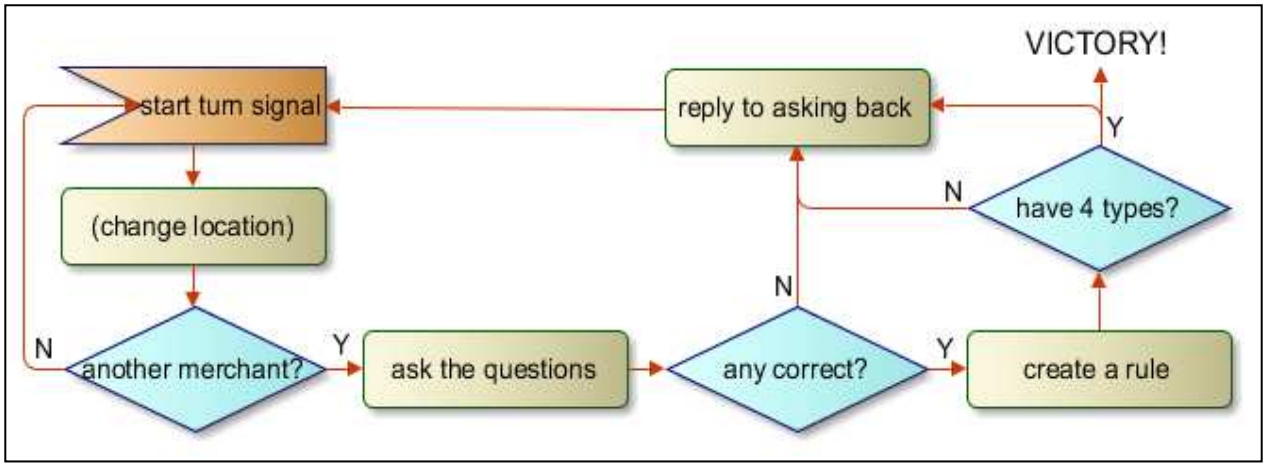

Figure 5-22. The game turn summary for one user

Keeping in mind that a merchant is to find goods from three other categories than the one of his good, for example a cider merchant needs to buy one sweet, one snack and one fruit, it gives at least 10 questions (2 questions per each of 5 turns, plus any asking back) to guess 3 times one of the needed 7(merchants)*6(goods)*5(places)=210 combinations. With some deduction and some luck, it is possible to achieve the goal. But as other merchants are also travelling in search for their required goods, the task is becoming even less obvious. This is where we hope the KRAMER system to intervene.

For each transaction made, the buyer is asked to introduce a corresponding rule into the COSMO application, for example: "the cake merchant in [a name of the place]". As discussed in Section 5.3, rules that are defined in COSMO are transmitted to the KRAMER server, which analyses the inserted situations and suggests important ones to other users. The same effect was anticipated in the test. Every successful transaction augments the collective knowledge, making it easier for other buyers. As a result, users were expected to receive suggestions on a situation defined by at least two other users (confidence threshold was set to 2 for the test), whenever a described situation re-occurred.

The suggestion notification was presented in a similar fashion to the normal one, with a slightly changed text and a special icon next to it (Fig. 5-23). Therefore, the aspects tested in this game were the ability to understand a received suggestion, and to make use of it in advancing in the game objective. In other words, we hoped to evaluate, whether providing new (novelty property) and pertinent (accuracy property) suggestions help users to acquire needed goods (utility property) and would make them to rely on the suggestions to assure the win (trust property). The conclusions were drawn based on the game observations and our questionnaire.

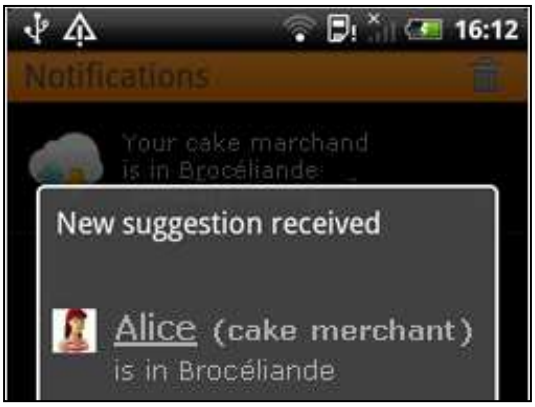

Figure 5-23. A game suggested notification

From the algorithmic point of view, situations processed by the KRAMER server are a subject of generalization. This effect was also simulated in the test, as there were some more abstract rules governing the game universe. In fact, every type of good is permitted to be purchased in a particular type of place. There are four general rules, listed in Table 5-4, to be discovered during the course of the game. Players are not conscious of them in the beginning, as their knowledge is limited to what they are selling and who is the second merchant of the same type of good (step 4 in the first game phase). The inducted knowledge becomes available as more and more situations are introduced into the system.

Table 5-4. Four abstract game rules

	Situation
#1	A merchant of drinks can sell his/her goods at the seaside.
#2	A merchant of sweets can sell his/her goods in the forest.
#3	A merchant of snacks can sell his/her goods on the plains.
#4	A merchant of fruits can sell his/her goods in the mountains.

In summary, game participants were expected to follow the KRAMER suggestions in order to achieve the goal in smaller amount of turns than by playing the same game without our system. Obviously, frequently a merchant's whereabouts, which one might got notified of, changed before a player needing his type of good could arrive on place in time. Moreover, as rules permitted three participants to perform their turns simultaneously, it happened sometimes on the very player's turn. As a result, a corresponding notification was disappearing in real time and a player was forced to interact with other merchants. This caused an entertaining effect that made the game really a game. Furthermore, it enabled rules introduced into the system to be diverse.

Our initial analysis of the test starts with the learning phase. There, all the instructions have been completed by all participants. Only testers of age above 60 years have met some troubles with the service and the notion of "rules", which required our intervention. One cause of it was that the prototype was not optimized in terms of the ergonomy. It certainly should be redesigned as far as defining important situations is concerned. The concept of notification rules is not a trivial one, but is critical in the KRAMER system. In summary, the learning phase permitted all testers but one to understand well the ideas of sharing context on different levels of abstractions, and creating rules launching a notification in a desired situation.

Familiarization with the COSMO application was only a prelude to the KRAMER system test. This part took a form of 30 minutes long games for both groups. As the game was based on a semi-cooperative scenario, let's focus on the test group, where all participants understood the learning phase. The game lasted 4 turns (even less than the limit of 5). In that time 5 out of 8 participants have reached the goal of having four types of goods. The players have entered 24 notification rules, among them 2 that were not correct, which resulted in 9 distinct situations introduced into the system.

Further analyses of the logs show that after a couple of players having luckily found early in the game that chips can be bought in Pleumeur-Bodou and in Brocéliande, after 14 minutes from the start all players have already bought chips somewhere on the plains, making snacks no longer needed by anyone. Other types of transactions were also reported in the course of the game. As a result, KRAMER was generalizing them into more abstract rules, which could be suggested to users more and more frequently. After four turns, the system found four abstract rules governing the game (Table 5-5). These are not exactly the same as those in Table 5-4, as some of them are still their specializations.

Table 5-5. Four abstract rules found in the game

	Situation
#1	A merchant of juice can sell his/her goods in Pleumeur-Bodou.
#2	A merchant of sweets can sell his/her goods in the forest.
#3	A merchant of chips can sell his/her goods on the plains.
#4	A merchant of fruits can sell his/her goods in the mountains.

The patterns in chasing goods show that some successful transactions were most probably an effect of following a suggestion received. However, only 7 questioned users of 8 in the group have noticed notifications for situations that they did not define themselves. Among those 7, only 5 report to have followed those suggestions, obtaining a total of 7 goods in this way (almost 2 goods each). In fact for those five testers, either one transaction was a result of luck or they were the ones not having finished the game with all four types of goods. All in all, we estimate that 32% of the successful transactions were inspired by the KRAMER system. One noticeable thing is that only one tester has reported to notice that a suggestion that she has received was a generalized one.

Taking into account both test groups, we might say that almost all users understood our system. Not only did they manage to learn new ideas and ways to manipulate them via a not optimized user interface of the COSMO application, but they seemed to appreciate the social aspects of the KRAMER system overall. Learning that context can be shared on different levels of abstractions, and on those different levels it can be defined as a notifiable situation is not a trivial thing to do. Nevertheless, some of the testers were able to employ the concept in a game environment, which was further

demanding as the notion of suggestions was not explained at all before the end of the test.

After the test we asked the participants if they found the KRAMER suggestions useful and whether they would like to use a context phonebook with a similar situation-based notification system in their environment. 13 out of the total 16 testers replied positively and gave a list of situations especially interesting for KRAMER. Most of them found the coordination for organizing an event as a situation that sharing location and/or availability status might prove useful. Knowing the whereabouts of others, especially their children and grand-children seemed appealing to others. Professional alerts were also mentioned as yet another use case. Finally, testers imagined wanting to know the availability of specialists, or open/closed state of places like restaurants.

5.3. Result discussion

Both of the performed tests, the mechanisms simulation and the user study, shed a light on the KRAMER system's properties enabling us to evaluate it. Among parameters listed in Table 5-1, there is a great deal that the two tests did cover. The most directly tested parameter in the simulation is the scalability of the system with respect to the number of rules contributed to the KRAMER server. Time charts in Section 5.1 approximate the computational complexity function of the generalization algorithm. From comparing several of the figures we learn that computations time depend mostly on the amount of context dimensions used in defining situations.

The shape of graphs of the function has a very peculiar, stairs-like nature. It appears that there are some border factors influencing the scalability function for its lowest arguments. However, after two such steps, the function becomes linear. We are able to estimate the bound limits for algorithm's computational complexity. Formulas (4) and (5) show the upper and lower bound limits respectively, in dependence of the number of input situations S and the maximum number of arcs in conceptual graphs describing them A.

$$\Theta(S^2A^4) \xrightarrow{A=const} \Theta(S^2) \qquad (4)$$

$$O(SA^2) \xrightarrow{A=const} O(S) \qquad (5)$$

More detailed analyses of the algorithm's performance show that the grouping part of the algorithm influences the time consumption in a much greater deal than the generalization part. Furthermore, the algorithm seems to perform closer to the lower

bound limit of the computational complexity (5), judging by the linear graphs. Particularly Fig. 5-18 promises to process 7000 rules in around 2.5 seconds, which is a good result for a first implementation algorithm of a recommender system. We say that the system scales in an acceptable fashion.

Regarding the coverage parameter, from the first test we learn that KRAMER operating on a completely covered set of data (all possible rules introduced) is not going to work properly. Fig. 5-2 and 5-4 show that not only the time of processing may be unacceptably long, but the system ceases to perform his main task of grouping and abstracting semantically similar situations. For this reason we assume having a number of situations that would actually have a meaning to users, which we called points of interest. Having covered all of those in the system would work fine. But the actual coverage and the number of point of interest depend on the users.

On the other side, knowing that KRAMER would not perform well if being injected with rules with all random situations tells us a lot about the robustness of the system. To understand why our system performs so badly in that case, one would need to look at the definition of similar situations that we adapted in Section 4.7.1. The first requirement for two situations to be considered as possibly similar was for them to have a matching graph structure, excluding the empty "any" concepts. The same applies in the generalization, which cannot have a different structure than the initial situations. Therefore, a situation like in Fig. 5-24 results in a cross combination of concepts for the output rules and the exponential output growth (Fig. 5-4).

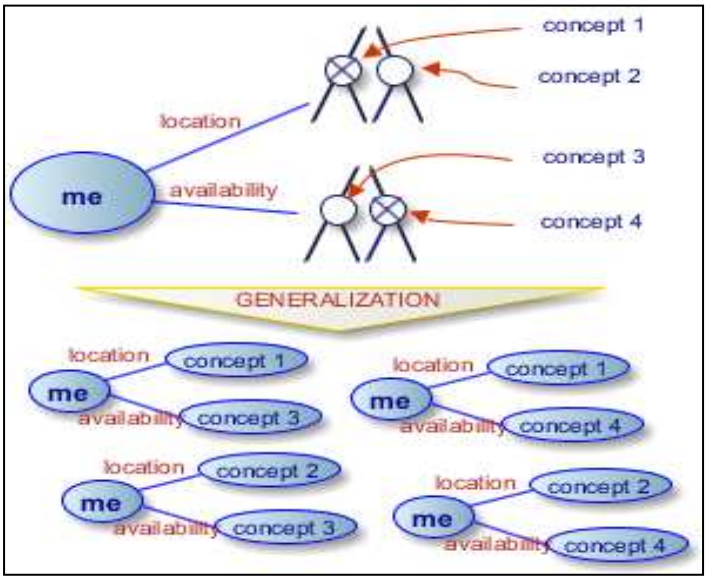

Figure 5-24. Cross combination of concepts in the generalization process

To avoid the problem, a mechanism for detecting that the system is being spammed with random rules would be required. This is because a system that does not forget any

of the input rules, even single representatives of such situations, would cause trouble. Therefore, maybe a mechanism forgetting older rules that were not reused any time recently could be implemented. This would also be an approach to introduce any kind of adaptivity to KRAMER. For the moment, this parameter is not considered in our prototype. Neither was it evaluated in either of the tests.

The second test, a user study in the form of a game, gave us more insight on user perception of the system, and the suggestions being provided. We tried to show our testers that the recommendations can be useful, especially in the context of winning a game before other participants do. Providing that the rules created were correct (the case for 22 rules out of 24), all of them were related to the goal of the game. Thanks to them, 5 out of 7 participants have followed the suggestions gathering almost 2/3 of their goods. In the whole test scale around 1/3 of successful transactions was done after receiving a recommendation.

During and after the test we asked our volunteers to evaluate different aspects of the system presented to them, including the interface of the COSMO application. We have gathered a long list of remarks and ideas, but what is more important, 13 out of 16 participants were interested in using a similar system in the future and gave KRAMER a positive note. Both the numbers from the previous paragraph and the questionnaire answers show that users see utility in KRAMER and are willing to trust it. However, they still have a limited picture of what a system like KRAMER could do, and in which situations it might prove useful.

Evaluating accuracy would be not fair after this test, as the suggestions were by definition accurate. The case is similar for parameters like novelty, serendipity or diversity. Those would require a long-term large-scale test, and a possibility for users to give a feedback on their impressions for particular suggestions. From the system's point of view, we have failed to evaluate the confidence parameter, as we were unable to experiment with the threshold for suggesting rules or not. We have no clear view of how this parameter should depend on the number of rules introduced into the system. Again, a large-scale test is required.

The final concern in any system, where users share their private data, is privacy, an ability of a system to assure that no unintended exploitation of this data would be done by third parties. A system processing personal situations is especially concerned as entities with malicious intentions could learn not only one's social graph structure, but also his whereabouts and those of his close ones. Fortunately, the KRAMER system

deals with the issue in its very fundaments. Noone accessing a KRAMER server could learn a true situation one finds himself in as the KRAMER system has no knowledge of the current user context, nor about the contacts he has. The action firing mechanism, which needs that information, is implemented locally in COSMO modules.

Moreover, sharing situations one wishes to fire an action upon does not compromise any private data. The very model of situation that we adapt assures that entities involved in defined situations are not described by names, phone numbers, or any other private data. The model uses relationship concepts, like "wife", "boss", "friend", *etc.* Wanting to be notified of a specific context dimension value of a friend does not carry any dangerously private data. The same cannot be said about the context distribution system, which KRAMER may rely on. In this case we imagine a set of mechanisms to let a user decide, what data she shares, like the abstraction levels definition as in our prototype. After all, people already share all sorts of private data in messengers and social networks. KRAMER just makes them be more useful.

Finally, it would take some highly unlike set of relations used in a rule for it to be associated with a person with such an unlike set of contacts. Assuring that the respective taxonomy relates to rather usual concepts, eliminates a threat of discovering someone's potentially embarrassing rule decisions. Moreover, for any situation to be suggested, it needs to gain enough popularity in the community, and therefore several similar situations need to be introduced by users, which makes the suggestions both more abstract and related to at least those several users. However, a more exhaustive study could tell us more about similar users groups that might be identified based on situations suggested. In the test game no-one has guessed other participant's identity this way.

In the questionnaire we have also asked for testers' appreciation for the COSMO module design and its ergonomics. It should be noted that the prototype has not been optimized in this respect, and this part does not concern directly performance of the KRAMER system, but this kind of feedback would prove useful should a large-scale test or commercialization be considered. As expected, among others, many testers have raised the subject of rule situations definition, see Fig. 4-18. The proposed set of simple dropdown lists filled with taxonomy concepts to choose from is not a user-friendly approach. However, no better solution has been proposed and this issue is left for further studies.

6. System applications

For the purpose of prototyping the KRAMER system we have decided to apply the latter to the domain of interpersonal communication. This prototype enabled us to validate the idea of having a situation-based recommender system. It also rendered evaluations of the system possible. However, the KRAMER system itself was designed to be generic, and we wanted to assure its easy future adaptations for a variety of domains, the one focusing on the social context of users being just one of them. There are several properties of KRAMER that stand for its genericity, while several aspects of it still need some adjustment for porting the system.

One huge advantage of the KRAMER system is the model of situations it adopts. Conceptual trees may be used for any kind of a situation, not only those involving context of friends and relatives. We have already introduced computation entities, like devices, services, applications, agents, *etc.* in Section 4.2. The composition of context of those connected things as well as human beings may constitute important situations in today's digital society. Moreover, the number and nature of context dimensions is not limited to one's location and activity status. In fact a situation graph may be of any desired complexity.

While operating on very complex situations may cause an issue as far as the ergonomy and design of the COSMO module are concerned, it has a minor impact on the situations processing mechanisms on a KRAMER server. Our generalization algorithm, Section 4.7, may deal just as well with both broader and deeper trees. One limitation would be assuring the introduction of any new context dimensions to both COSMO and KRAMER. Each of them needs to be aware of the corresponding taxonomies, and share knowledge on any changes to either one of them. The KRAMER system in general needs to maintain common semantics.

The architecture of the system proposed in Section 4.3 in general case could keep the client-server nature. Though it would not necessarily use a GSM network, as in Fig. 4-8. It might also introduce a number of new elements, like home gateways aggregating a set of context-aware home appliances, or other devices processing sensor networks data into more abstract virtual context dimensions. The COSMO module would still play the role of an interface with users for managing important situations, both created and suggested. But the core of the system, the KRAMER server along with its interfaces, would stay intact.

Finally, further research would need to be done to estimate the impact of introducing unrestricted actions to be fired upon situation occurrence. By limiting our prototype to fire notifications only, we have not treated the problem of situations similarity in the system with respect to the associated actions. Should two different actions for one situation cause the two rules to be treated separately, or rather force the actions to be generalized as well, these are a couple of questions we will try to find answers for in the future.

The following subsections focus on two application domains considered by us, listing the associated advantages. Please note that this is not an exhaustive list of KRAMER possible uses. Any entity sensing either internal or external context can be incorporated into an important situation, making the action taking API the only limitation of exploiting the collective power, the collective intelligence of which is provided by the KRAMER. In particular, any mix of such entities and their principal domains is also possible.

6.1. Social computing

Our initial choice for targeting with KRAMER social behaviours such as interpersonal communication was due to anticipation that the system would have valuable social computing features. It supports several natural human needs as far as information exchange in communities is concerned. Those are namely a need to share information with others, a desire to stay up to date with new fact about others, an influence that a community has over each of its member's information perceiving and over their making corresponding decisions.

The KRAMER system in its current shape relies on a system sharing different context dimensions between its users. We see this practice an adaptation of microblogging phenomena, which is an already popular mode of communication for people [JAV07]. Those users not only send messages to one another, messages that disclose a great deal of private information one might add, but they started to broadcast it to many other people. Tools like social networks enable publishing posts so that all friends, or even all subscribed users could read, and further comment.

At least in some cases, such messages aim at actually delivering pieces of information to others. There are many social reasons for doing so, reassuring, maintaining closeness, and inspiring an informed decision being just three of them. We argue that one's situation is important information in social interactions. The context of

another person one is in contact with is a naturally present feature in classical interactions. If the technology is to support such natural circumstances of human behaviours, the context should play a major role in communication.

We make the context sharing process a further socially meaningful act of interaction, as the data can be disclosed on different levels of details for different contacts. Being shared with but a little portion of details might be considered as a signal that one is not considered to be close and important. As a result, he could revaluate his levels. Therefore, the adjustment of COSMO sharing preferences is a social negotiation process in coherence with the real life interpersonal relations. It is also a part of an indirect communication.

While making a piece of context visible to others is an obvious interaction, consulting this data by others is an interaction as well. Even though it might not be acknowledged by the context owner, but it is like asking a question: "I wonder, what my husband is doing now?", with a system replying: "He wanted me to tell you that he's driving a car". Humans want to learn situations of other people for the same reasons that those people anticipated sharing them in the first place: taking decisions, being reassured, *etc.*

However, an application like phonebook is not something one consults every five minutes, whereas it might be required to stay up to date. On the other hand, one having communicated his private piece of data may be hoping for some kind of social feedback, as he would get in face to face interactions. Therefore we enable defining important situations in the scope of action firing rules. Every situation defined in KRAMER is a socially enabled one, because it may involve the context of other contacts. Furthermore, a decision to fire an action or not is dependant on the levels of details of the context received.

Finally, the KRAMER system introduces a collective way for a community to determine important situations. It is known from [SUR05] and [ALA08] that the whole communities can be empowered from both collaboration and competition of their members. Technologies like collaborative filtering [SAR01] emulate a huge amount of information exchange, but making it anonymous. The algorithms associated can compute a collective intelligence, which is greater than the one of the smartest individual. KRAMER is a recommender system based on this principle, which in this application domain is facilitating further the discussed social interactions.

6.2. Home automation

The domain of smart homes (or even smart cities) is directly related to “pervasive computing”, which is Mark Weiser's model of human seamless interaction with objects embedded with computational devices [WEI97]. With the technologies like sensor networks [AKY02] or Internet of Things [ATZ10] getting more and more attention, and the necessary hardware becoming available at relatively low cost, smart surroundings are expected to revolutionize our lives really soon. It is therefore not surprising that researchers already design platforms for smart home automation, [MER08] being one example of.

Generally, pervasive computing enables the connected computing entities to cooperate in performing tasks that normally humans needed to do themselves. Devices embedded in objects like lamps or windows not only sense information about the objects' context, but may have also access to their execution functions, like switching on/off or opening/closing. This opens a huge opportunity to program particular tasks to be performed automatically in given situations. Home intelligence would be made of such smart applications of a simple rule-based nature: `DO <action> WHEN <situation>`.

Considering such rules as a base element of intelligent homes, makes the latter a perfect domain for the KRAMER system. Researchers have already considered the need to export one's applications for his smart home to enable their instant mapping on a different set of objects [CHE09]. This cross house portability of applications is based there on description files created for a given user, so that her preferences are preserved in different environments. In consequence, such applications would adapt their functionality opportunistically according to objects found after the mobility.

The KRAMER system would go another step further. It would render possible both determining useful applications and sharing them in a community of users. In that sense not only would it be cross house adaptive, but further “cross similar user” portable. A dimension of sharing applications in the scope of a community would make them a subject of KRAMER server processing. And like-wise, it would harvest those applications, which are considered useful widely among community members, so that they could be suggested to those, who did not define them themselves.

To give an example of such an application, let's consider the following rule “when TV set is switched on, and blender is on, turn up the TV volume”. This is a simple

application introducing a possibly useful functionality of adjusting the volume of a TV set in function of the noise of a blender. There are evidently two parts of the rule present, which are known from the KRAMER: the condition and the result. The condition is a composition of values for context dimensions, which ubiquitous devices are likely to be able to sense. The result is an action, assuming existence of a TV set providing an API to modify the sound volume.

Visibly, the situations in this application domain do not differ a lot from the ones adapted in our prototype. Instead of social situations, there would be states of home environment. Instead of activity-related availability statuses, there would be internal device states. Instead of interpersonal relations, there would be types of objects. KRAMER could very well maintain the same situation model and be only introduced with new taxonomies. Those semantics providing structures would enable the KRAMER server algorithm to perform its generalization on new context dimensions.

One main challenge to be considered is how the system, *i.e.* the aggregation algorithm, would deal with the application actions. Considering a set of simple and predefined action concepts, having two different outcomes of similar situations, or one particular outcome common for two different situations in two applications could either influence the similarity of those situations or not. Moreover, actions available in pervasive environments could be also modelled by a corresponding taxonomy, enabling them to be defined on different levels of abstractions, like in Pobicos [LAL10]. Finally, coexistence of several applications might result in their conflicting logics as explained in [NHL08]. All these issues need to be taken into account while adapting KRAMER to home automation domain.

7. Final thoughts

In this dissertation we discussed our research results on a collaborative situation-aware system, KRAMER. Our system learns which situations are important for people in the community of its users. By enabling the users to associate such situations with firing particular actions, the knowledge the system acquires is a collection of user-defined rules of a type `DO <action> WHEN <situation>`. The data is then processed by the KRAMER to derive collective intelligence, which is in turn shared with all community members in the form of suggestions. The system suggests taking particular actions while facing given important situations.

KRAMER is a recommender system based on a collaborative filtering principle. Unlike in usual collaborative filtering implementations, the object of recommendations is not a simple item, like a book or movie, but a pair of a complex situation and an action. This requires our solution to apply semantic technology to processing suggestion objects, instead of purely statistic calculations. As a result KRAMER users are to be empowered with situation-aware recommendations in their decision making on taking actions.

Therefore, in this thesis we give an answer to the question we asked ourselves when starting this research: how can a system adapt natural social interactions and enable one to profit from collaborative knowledge while encountering new situations, which would potentially require a corresponding decision. The KRAMER system is one technical solution to a problem of combining situation awareness and collective intelligence technologies. While the system is completely generic its prototype is applied to social communication scenario. But the KRAMER system may be adapted to any domain that deals with the problem expressed as a set of conceptual graphs associated with an action firing logic, where the solution would be extracted from the respective data gathered in a collaborative fashion.

Having defined, implemented and tested our system we are ready to evaluate the initial assumptions listed in Table 3-1. Regarding ***A1***, there is indeed collective knowledge coming from the composition of many user situation-related experiences. An intelligence that can be derived of it may not only enhance social communication services as stated in ***A2***, but further empower situation-aware decision making and support any kind of rule-based smart systems. Finally, as situations processed in the scope of the system are conceptual representations of context data describing the

surrounding and social circumstances, it is vital to employ semantic technology, which supports ***A3***.

In the process of designing our collaborative system, it has acquired a certain number of unique properties. For example, the KRAMER system adapted to interpersonal communication domain is a social medium with strong social computing support. Moreover, several aspects of it, as well as some methodology approaches that we tackled, constitute our contribution to science. Table 7-1 present a list of those contributions.

Table 7-1. Our contributions

C1. *Use of collaborative filtering in KRAMER enriching situation awareness field*
C2. *Model of situations defined in KRAMER as semantic conceptual trees*
C3. *Manipulations on situations facilitated in KRAMER thanks to situation lattices*
C4. *Algorithm generalizing sets of conceptual trees in KRAMER proposed*
C5. *Authorial game-based evaluation of KRAMER for a small scale test*

While it was possible to draw strong conclusions from the research presented in this dissertation, there are several elements still missing to make our work more complete. First of all, we have managed to simplify the prototype of KRAMER by limiting the possible actions to just notifications. It did make sense for a social communication scenario, where users are always in charge of their actions, which are not in turn restricted after having received a notification. We admit, however, that in the home automation variant, for instance, actions could be programmed to fire automatically, which would affect the system for reasons stated in Section 6.2.

Moreover, we have also avoided mixing our semantic-centered collaborative filtering with the classical statistically-oriented one. In particular, it might be beneficial to not simply group situations by their semantic similarity, but further differentiate them with respect to which users defined them. Collaborative filtering is often said to connect like-minded users, as receiving a suggestion of an item well evaluated by one user depends on evaluation similarity on other items evaluations. In our case it might happen that not all community members have similar vision on important situations, and suggestion of the latter would be more accurate should they be compared with respect to user similarity.

We regret also not to have a possibility to perform a large-scale user test of KRAMER. One would expect to deepen the system's evaluations by having it tested for several months on a numerous group of users. For instance, more insight would be taken on the parameters like coverage and confidence. With bigger expected number of rules introduced we could observe a real use of the system. As a result, we could experiment with introducing user-to-user similarity and comparing the results with and without that notion. We would obtain much more data for further off-line analyses, with one very important: how would a real usage evolution of the system look like in function of time. There is a great chance for performing such a test, as the KRAMER system will be reused in one of the Orange Labs projects following this research.

Furthermore, we have already defined several future expansions for the system. Adapting KRAMER for further domains of application would seem the most evident one. Any environment with entities able to sense their context and open to executing actions on them is suitable. In Section 6.2 we introduce the domain of home appliances, which would benefit from portable smart application suggestions. The technology of web services and their composition [ZHA09] is also tempting, but would require further study, as web services are *a priori* stateless. Any mix of such domain would be also interesting.

One other enhancement that we consider is a possibility to operate on situations slightly more complicated in terms of temporal relations. Being able to differentiate different elements of a situation as happening in a particular order, simultaneously or within any other temporal relation from [ALL83] might further augment the number of use cases for the system. The reason for not taking this aspect yet into implementation is its negative impact on readability of the COSMO module interface. Ergonomics are already an issue for defining composed situation, and would require a redesigning effort should the notion of time be introduced.

List of publications

1. BOUABDALLAH Ahmed, TOUTAIN François, SZCZERBAK Michał, and BONNIN Jean-Marie: **On the benefits of a network-centric implementation for context-aware telecom services**, ICIN 2011, *15th International Conference on Intelligence in Next Generation Networks*, 4-7 October 2011, Berlin, Germany, pp. 236-240.

2. SZCZERBAK Michał, TOUTAIN François, BOUABDALLAH Ahmed, and BONNIN Jean-Marie: **Collaborative Context Experience in a Phonebook**. WAINA 2012, *26th International Conference on Advanced Information Networking and Applications Workshops*, 26-29 March 2012, Fukuoka, Japan, pp. 1275-1281

3. SZCZERBAK Michał, BOUABDALLAH Ahmed, TOUTAIN François, and BONNIN Jean-Marie: **Generalizing contextual situations**. ICSC 2012, *6th IEEE International Conference on Semantic Computing*, IEEE, 19-21 September 2012, Palermo, Italy, pp. 293-301

4. SZCZERBAK Michał, BOUABDALLAH Ahmed, TOUTAIN François, and BONNIN Jean-Marie: **A model to compare and manipulate situations represented as semantically labelled graphs**. ICCS 2013, *20th International Conference on Conceptual Structures*, Ed. Springer-Verlag, 10-12 January 2013, Mumbai, India, 2013, vol. 7735 - Lecture Notes in Computer Science, pp. 44-57.

5. SZCZERBAK Michał, TOUTAIN François, BOUABDALLAH Ahmed, and BONNIN Jean-Marie: **KRAMER - New Social Medium Based on Collaborative Recognition of Important Situations**. *The Computer Journal*, Special Issue on Social Computing, in press.

List of patents

1. Procédé et système de notification, à un utilisateur d'un terminal, de données contextuelles relatives à des éléments identifiés dans une application de type répertoire (France)

References

[ACZ88] Aczel, P. (1988) **Not-Well-Founded Sets**. *CSLI Lecture Notes Nr 14*, Center for the Study of Language and Information, Stanford.

[ADO11] Adomavicius, G. and Tuzhilin, A. (2011) **Context-Aware Recommender Systems**. *Recommender Systems Handbook* by Ricci, F. *et al.*, Springer, pp. 217-253.

[AKM96] Akman, V., and Surav, M. (1996) **The Use of Situation Theory in Context Modeling**. *Computational Intelligence*, vol. 13(3), pp. 427-438.

[AKY02] Akyldiz, I.F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002) **Wireless Sensor Networks: A Survey**. *Computer Networks* Journal, vol. 38(4), pp. 393–422.

[ALA09] Alag, S. (2008) **Collective Intelligence in Action**. ISBN 978-1-933988-31-3, Manning.

[ALL83] Allen, J.F. (1983) **Maintaining Knowledge about Temporal Intervals**. *Communications of the ACM*, vol. 26(11), pp. 832-843.

[ANA06a] Anagnostopoulos, C., Ntarladimas, Y., and Hadjiefthymiades, S. (2006) **Reasoning about Situation Similarity**. *International IEEE Conference on Intelligent Systems*, pp. 109-114.

[ANA06b] Anagnostopoulos, C., Ntarladimas, Y., and Hadjiefthymiades, S. (2006) **Situation Awareness: Dealing with Vague Context**. *ACS/IEEE International Conference on Pervasive Services*, pp.131-140.

[ATZ10] Atzori, L., Iera, A., and Morabito, G. (2010) **The Internet of Things: A Survey**. *Computer Networks* Journal, vol. 54(15), pp. 2787-2805.

[BAL07] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007) **A Survey on Context-Aware Systems**. *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, pp. 263-277.

[BAR09] Baralis, E., Cagliero, L., Cerquitelli, T., Garza, P., and Marchetti, M. (2009) **Context-Aware User and Service Profiling by Means of Generalized Association Rules**. *13th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES'09)*, LNCS 5712, pp. 50-57.

[BAR08] Barkhuus, L., Brown, B., Bell, M., Hall, M., Sherwood, S., and Chalmers, M. (2008) **From Awareness to Repartee: Sharing**

Location within Social Groups. *SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*, pp. 497-506.

[BAR83] Barwise, J. and Perry, J. (1983) **Situations and Attitudes**. *Bradford Books*, The MIT Press, ISBN 0-262-02189-7.

[BEI03] Beigl, M., Krohn, A., Zimmer, T., Decker, C., and Robinson, P. (2003) **AwareCon: Situation Aware Context Communication**. *Ubiquitous Computing*, Lecture Notes in Computer Science, vol. 2864, pp 132-139.

[BOL07] Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F.A., and Tanca, L. (2007) **A Data-Oriented Survey of Context Models**. *ACM SIGMOD Record*, vol. 36(4), pp 19-26.

[BRO07] Brown, B., Taylor, A.S., Izadi, S., Sellen, A., Kaye, J., and Eardley, R. (2007) **Locating Family Values: A Field Trial of the Whereabouts Clock.** *9th International Conference on Ubiquitous Computing (UbiComp'07)*, pp. 354-371.

[CAO06] Cao, Y., and Li, Y. (2006) **An Intelligent Fuzzy-Based Recommendation System for Consumer Electronic Products**. *Expert Systems with Applications*, vol. 33(1), pp. 230-240.

[CHE02] Chen, G., and Kotz, D. (2002) **Context aggregation and dissemination in ubiquitous computing systems**. *Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pp. 105-114.

[CHE05] Chen, A. (2005) **Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment.** *Location and Context-Awareness Lecture (LoCA'05)*, pp. 244-253.

[CHE09] Cheng, S.T., Wang, C.H., and Chen, C.C. (2009) **An Adaptive Scenario Based Reasoning System Cross Smart Houses**. *9th International Symposium on Communications and Information Technology (ISCIT'09)*, pp. 549-554.

[COO91] Cooper, R., and Kamp, H. (1991) **Negation in Situation Semantics and Discourse Representation Theory**. *Situation Theory and Its Applications vol. 2*, Stanford University.

[COS06] Costa, P.D., Guizzardi, G., Almeida, J.P.A., Pires, L.F., and van Sinderen, M. (2006) **Situations in Conceptual Modeling of Context**.

10th IEEE International Enterprise Distributed Object Computing Conference Workshops, p. 6.

[CRO07] Croitoru, M., Hu, B., Dashmapatra, S., Lewis, P., Dupplaw, D., and Xiao, L. (2007) **A Conceptual Graph Based Approach to Ontology Similarity Measure**. *15th International Conference on Conceptual Structures: Knowledge Architectures for Smart Applications*, pp. 154-164.

[DEL12] Delaveau, L., Loulier, B., Matson, E.T., and Dietz, E. (2012) **A vector-space retrieval system for contextual awareness**. *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pp. 162-165.

[DEV91] Devlin, K.J. (1991) **Situations as Mathematical Abstractions**. *Situation Theory and Its Applications vol. 2*, Stanford University.

[DEY00a] Dey, A.K. (2000) **Providing architectural support for building context-aware applications**. *PhD thesis*, Georgia Institute of Technology.

[DEY00b] Dey, A.K. and Abowd, G.D. (2000) **CybreMinder: A Context-Aware System for Supporting Reminders**. *2nd international symposium on Handheld and Ubiquitous Computing (HUC'00)*, pp. 172-186.

[DUC09] Ducheneaut, N., Partridge, K., Huang, Q., Price, B., Roberts, M., Chi, E.H., Belotti, V., and Begole, B. (2009) **Collaborative Filtering Is Not Enough? Experiments with a Mixed-Model Recommender for Leisure Activities.** *User Modeling, Adaptation, and Personalization*, LNCS, vol. 5535, pp. 295-306.

[EAG09] Eagle, N., Pentland, A., and Lazer, D. (2009) **Inferring Social Network Structure using Mobile Phone Data**. *Proceedings of the National Academy of Sciences (PNAS)*, vol. 106(36), pp. 15274-15278.

[ELL92] Ellis, G., and Levinson, R. (1992) **Multi-Level Hierarchical Retrieval**. *Knowledge-Based Systems, Conceptual Graphs Special Issue*, vol. 5, pp. 233-244.

[END95] Endsley, M.R. (1995) **Toward a Theory of Situation Awareness in Dynamic Systems**, *Human factors*, Vol. 37, pp. 32-64.

[ETT06] Etter, R., Dockhorn Costa, P., and Broens, T. (2006) **A Rule-Based Approach Towards Context-Aware User Notification Services**. *ACS/IEEE International Conference on Pervasive Services*, pp. 281-284.

[ETZ10] Etzion, O., and Niblett, P. (2010) **Event Processing in Action**. ISBN 1935182218-9781935182214, Manning.

[FLE09] Fleder, D. and Hosanagar, K. (2009) **Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity**. *Management Science*, vol. 55, pp. 697-712.

[FOR82] Forgy, C.L. (1982) **Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem**. *Artificial Intelligence*, vol. 19, pp. 17-37.

[GAN08] Gandon, F. (2008) **Graphes RDF et leur Manipulation pour la Gestion de Connaissances**, Chapter 4: Graphes comme espaces métriques. *HdR*, Nice Sophia-Antipolis.

[GRU07] Gruber, T. (2007) **Collective Knowledge Systems: Where the Social Web Meets the Semantic Web**. *Journal of Web Semantics*, vol. 6, pp. 4-13.

[GU09] Gu, T., Wu, Z., Tao, X., Pung, H.K., and Lu, J. (2009) **epSICAR: An Emerging Patterns based approach to sequential, interleaved and Concurrent Activity Recognition.** *IEEE International Conference on Pervasive Computing and Communications (PerCom'09)*, pp. 1-9.

[HAY85] Hayes-Roth, F. (1985) **Rule-based systems**. *Communications of the ACM*, vol. 28, pp.921-932.

[HER04] Herlocker, J.L., Konstan, J.A., Terveen, L.G., and Riedl, J.T. (2004) **Evaluating Collaborative Filtering Recommender Systems**. *ACM Transactions on Information Systems*, vol. 22, pp.5-53.

[HIL95] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995) **Recommending and evaluating choices in a virtual community of use**. *SIGCHI Conference on Human Factors in Computing Systems (CHI'95)*, pp.194-201.

[HON08] Hong, J., Suh, E.H., Kim, J., and Kim S.Y. (2008) **Context-Aware System for Proactive Personalized Service Based on Context History**. *Expert Systems with Applications*, vol. 36, pp. 7448-7457.

[JAV07] Java, A., Finin, T., Song, X., and Tseng, B. (2007) **Why We Twitter: Understanding Microblogging Usage and Communities**. *9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pp. 56-65.

[JIA08] Jiang, X., and Bunke, H. (2008) **Graph Matching**. *Studies in Computational Intelligence*, vol. 73, pp. 149-173.

[KNO08] Knox, S., Shannon, R., Coyle, L., Clear, A.K., Dobson, S., Quigley, A.J., and Nixon, P. (2008) **Scatterbox: Context-aware message management**. *Revue d'Intelligence Artificielle*, vol. 22(5), pp. 549-568.

[KOK09] Kokar, M.M. (2009) **Ontology-Based Situation Awareness**. *Information Fusion*, Special Issue on High-level Information Fusion and Situation Awareness, vol. 10(1), pp. 83-98.

[KOR04] Korpipää, P., Häkkilä, J., Kela, J., Ronkainen, S., and Känsälä, I. (2007) **Utilising context ontology in mobile device application personalisation**. *3rd international conference on Mobile and ubiquitous multimedia (MUM'04)*, pp. 133-140.

[LAL10] Lalis, S., Domaszewicz, J., Pruszkowski, A., Paczesny, T., Ala-Louko, M., Taumberger, M., Georgakoudis, G., and Lekkas, K. (2010) **Tangible Applications for Regular Objects: An End-User Model for Pervasive Computing at Home**. *The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'10)*, pp. 385-390.

[LAU07] Laudy, C., Ganascia, J.G., and Sedogbo, C. (2007) **High-level Fusion based on Conceptual Graphs**. *10th International Conference on Information Fusion*, pp. 1-8.

[LED03] Lederer, S., Mankoff, J., and Dey, A.K. (2003) **Who Wants to Know What When? Privacy Preference Determinants in Ubiquitous Computing**. *Extended Abstracts on Human Factors in Computing Systems (CHI'03)*, pp. 724-725.

[LEE10] Lee, A. (2010) **Exploiting context for mobile user experience**. *Proceedings of the First Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS)*, ACM Press.

[LEV92] Levinson, R., and Ellis, G. (1992) **Multilevel Hierarchical Retrieval**. *6th Annual Conceptual Graphs Workshop*.

[MAK05] Mäkelä, E., (2005) **Survey of semantic search research**. *Seminar on Knowledge Management on the Semantic Web*, Department of Computer Science, University of Helsinki.

[MAR02] Marcos, E., and Cavero, J.M. (2002) **Hierarchies in Object Oriented Conceptual Modeling**. *Advances in Object-Oriented Information Systems Workshop*, Lecture Notes in Computer Science, vol. 2426, pp. 24-33.

[MCC07] McCall, J.C., and Trivedi, M.M. (2007) **Driver Behavior and Situation Aware Brake Assistance for Intelligent Vehicles**. *Proceedings of the IEEE*, vol. 95(2), pp. 374–387.

[MEC07] Mechkour, S. (2007) **Overview of Situation Theory and its application in modeling context**. *Seminar paper*, Univ. of Fribourg.

[MEI04] Meissen, U., Pfennigschmidt, S., Voisard, A., and Wahnfried, T. (2004) **Context-and Situation-Awareness in Information Logistics**. *Current Trends in Database Technology - EDBT2004 Workshops*, pp. 448-451.

[MER08] Merabti, M., Fergus, P., Abuelma'atti, O., Yu, H., and Judice, C. (2008) **Managing Distributed Networked Appliances in Home Networks**. *Proceedings of the IEEE*, vol. 96(1), pp. 166-185.

[MON00] Montes-y-Gómez, M, Gelbukh, A., and López-López, A. (2000) **Comparison of Conceptual Graphs**. *Mexican International Conference on Artificial Intelligence*, pp. 548-556.

[MON01] Montes-y-Gómez, M., Gelbukh, A., López-López, A., and Baeza-Yates, R. (2001) **Flexible Comparison of Conceptual Graphs**. *12th International Conference on Database and Expert Systems Applications*, pp. 102-111.

[MUG93] Mugnier, M.L. (1995) **On Generalization / Specialization for Conceptual Graphs**. *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 7, pp. 325-344.

[MUN10] Muñoz-Organero, M., Ramírez-González, G. A., Muñoz-Merino, P. J., and Kloos, C. D. (2010) **A Collaborative Recommender System Based on Space-Time Similarities**. *IEEE Pervasive Computing Magazine*, Vol. 9(3), pp. 81-87.

[NAU10] Naudet, Y., Schwartz, L., Mignon, S., and Foulonneau, M. (2010) **Applications of User and Context-Aware Recommendations Using**

Ontologies. *Conference Internationale Francophone sur l'Interaction Homme-Machine (IHM'10)*, pp. 165-172.

[NHL08] Nhlabatsi, A., Laney, R., and Nuseibeh, B. (2008) **Feature Interaction: The Security Threat from Within Software Systems**. *Progress in Informatics*, Special Issue on The Future of Software Engineering for Security and Privacy, vol. 5, pp. 75-89.

[PAD04] Padovitz, A., Loke, S.W., and Zaslavsky, A. (2004) **Towards a theory of context spaces**. *Second IEEE Annual Conference onPervasive Computing and Communications Workshops*,pp 38-42.

[PAZ07] Pazzani, M., and Billsus, D. (2007) **Content-Based Recommendation Systems**. *The Adaptive Web*, LNCS 4321, pp. 325-341.

[POO95] Poole, J., and Campbell, J.A. (1995) **A Novel Algorithm for Matching Conceptual and Related Graphs**. *3rd International Conference on Conceptual Structures: Applications, Implementation and Theory*, pp. 293-307.

[RAE05] Raento, M., Oulasvirta, A., Petit, R., and Toivonen, H. (2005) **Contextphone: A Prototyping Platform for Contxt-Aware Mobile Applications**. *IEEE Pervasive Computing Magazine*, Vol. 4(2), pp. 51-59.

[REE05] Reed, R.N., and Kocura, P. (2005) **Conceptual Graph based Criminal Intelligence Analysis**. *International Conference on Conceptual Structures*, pp. 146-159.

[ROB08] Roberts, M., Ducheneaut, N., Begole, B., Partridge, K., Price, B., Bellotti, V., Walendowski, A., and Rasmussen, P. (2008) **Scalable Architecture for Context-Aware Activity-Detecting Mobile Recommendation Systems**. *ADAMUS Workshop at IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, pp. 1-6.

[SAR01] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001) **Item-based collaborative filtering recommendation algorithms**. *10th International Conference on World Wide Web (WWW'01)*, pp. 285-295.

[SAR10] Saruladha, K., Aghila, G., and Raj, S. (2010) **A Survey of Semantic Similarity Methods for Ontology Based Information Retrieval**. *2nd International Conference on Machine Learning and Computing (ICMLC)*, pp. 297-301.

[SCH01a] Shafer, J.B., Konstan, J., and Riedl, J. (2001) **E-commerce Recommendation Applications**. *Data mining and knowledge discovery*, vol. 5, pp. 115-153.

[SCH01b] Schmidt, A., Stuhr, T., and Gellersen, H. (2001) **Context-Phonebook – Extending Mobile Phone Applications with Context**. *Third International Workshop on Human Computer Interaction with Mobile Devices (Mobile HCI'01)*.

[SCH11] Shani, G. and Gunawardana, A. (2011) **Evaluating Recommendation Systems**. *Recommender Systems Handbook* by Ricci, F. *et al.*, Springer, pp. 257-297.

[SOW83] Sowa, J. (1983) **Conceptual Structures: Information Processing in Mind and Machine**. *Ed. Addison-Wesley*.

[SOW08] Sowa, J. (2008) **Conceptual Graphs**. *Foundations of Artificial Intelligence*, vol. 3, pp. 213-237.

[STO11] Stojanovic, N., Stojanovic, L., Anicic, D., Ma, J., Sen, S., and Stuhmer, R. (2011) **Semantic complex event reasoning – beyond complex event processing**. *Foundations for the Web of Information and Services*, Springer-Verlag Berlin Heidelberg, pp. 253-278.

[STR04] Strang, T., and Linnhoff-Popien, C. (2004) **A Context Modeling Survey**. *First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp'04*.

[SUR05] Surowiecki, J. (2005) **The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations**. *Anchor*, ISBN 978-0385721707.

[TOU11] Toutain, F., Bouabdallah, A., Zemek, R., and Daloz, C. (2011) **Interpersonal Context-Aware Communication Services**. *IEEE Communications Magazine*, vol. 49(1), pp. 68-74.

[WAN04] Wang, X.H., Zhang, D.Q., Gu, T., and Pung, H.K. (2004) **Ontology Based Context Modeling and Reasoning Using OWL**. *Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pp. 18-22.

[WAN07] Wang, F.Y., Zeng, D., Carley, K.M., Mao, W. (2007) **Social computing: from social informatics to social intelligence**. *IEEE Computer Society*, vol. 22, pp. 79-83.

[WEI97] Weiser, M., and Brown, J.S. (1997) **The Coming Age of Calm Technology**. *Beyond calculation*, ISBN: 0-38794932-1, pp. 75-85.

[YAU06] Yau, S.S., and Liu, J. (2006) **Hierarchical Situation Modeling and Reasoning for Pervasive Computing**. *4th IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, pp. 5-10.

[YE07] Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2007) **Using Situation Lattices to Model and Reason about Context**. *4th International Workshop on Modeling and Reasoning in Context*, pp. 1-12.

[YE08] Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2008) **Representing and Manipulating Situation Hierarchies using Situation Lattices**. *Modelling and Reasoning on Context*, pp. 647-667.

[YE09] Ye, J., and Dobson, S. (2009) **Human Behaviour Study with Situation Lattices**. *IEEE International Conference on Systems, Man and Cybernetics*, pp. 343-348.

[ZAI02] Zaïane, O.R. (2002) **Building a Recommender Agent for e-Learning Systems**. *International Conference on Computers in Education (ICCE'02)*, pp. 55-59.

[ZHA09] Zhao, Z., Laga, N., and Crespi, N. (2009) **A Survey of User Generated Service**. *IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC'09)*, pp. 241-246.

[ZHE09] Zheng, Y., Zhang, L., Xie, X., and Ma, W.Y. (2009) **Mining Interesting Locations and Travel Sequences from GPS Trajectories**. *18th International Conference on World Wide Web*, pp. 791-800.

[ZHO02] Zhong, J., Zhu, H., Li, J., and Yu, Y. (2002) **Conceptual Graph Matching for Semantic Search**. *10th International Conference on Conceptual Structures: Integration and Interfaces (ICCS '02)*, pp. 92-106.