ORCHESTRA: Formalism to Express Static and Dynamic Model of Mobile Collaborative Activities and Associated Patterns

Bertrand David, René Chalon, Olivier Delotte, Guillaume Masserey

LIESP Laboratory
Ecole Centrale de Lyon
36, av. Guy de Collongue, 69134 Ecully Cedex, France
Bertrand.David@ec-lyon.fr

Abstract. Orchestra is a new formalism on which we are working in the field of cooperative systems design. In CoCSys methodology for Cooperative Capillary Systems design, we transform partial scenarios describing particular cooperative situations in a more comprehensive Cooperative Behaviour Model (CBM). In this paper, we describe our contribution to the need for a graphical formalism which would be able to express in a natural way, understandable by different actors (users, designers, developers,…) different cooperation situations in an ambient intelligence environment (mobile, context-aware, proactive and ubiquitous). ORCHESTRA is complementary to CTT and UML Use cases, and its objective is to express clearly cooperation situations (explaining easily synchronous or asynchronous cooperation activities) and the role (active or passive) played instantaneously by each actor. We take into account main concepts of "cooperative world" which are Actors, Roles, Groups, Tasks, Processes, Artefacts (Tools and Objects) and Contexts (Platforms, Situations and Users). With Orchestra formalism we try to express by a sort of music staff individual and collective behaviours. In this way we can model either individual works or organized collective activities. We present this formalism, its metamodel and associated patterns expressing typical configurations of cooperation facilitating their reuse.

Keywords: CSCW, Specific Description Language, MDA inspired elaboration process, transformation process, formalism meta-model, description patterns

1 Introduction

CSCW [2] is a field of interactive computer-based systems which objective is to allow several participants (actors) to work together via a computer-based system to complete cooperatively a task which can be of different natures (design, management, production, learning, etc). Design of this kind of systems is relatively complex because it is not limited to individual activities, but also and mainly to cooperative

work of several actors, which can be classified in co-operation, coordination and conversation activities in respect with the definition initially proposed by Ellis [10] and adapted by several other authors [8]. This cooperative work can be done in several cooperative situations characterized initially by Johansen and enhanced by Ellis [9]. At the moment CSCW systems are becoming more and more mobile, context-aware and proactive. We called this kind of cooperative systems Capillary Cooperative Systems (CCS) [6]. We use this term by analogy with the network of blood vessels. The purpose of the Capillary CS is “to extend the capacities provided by co-operative working tools in increasingly fine ramifications, hence they can use fixed workstations and handheld devices". These systems become also pervasive, proactive and ubiquitous. Our final goal is to allow them to evolve in mixed reality environment (mixture of real and digital objects and tools) and to put into practice Ambient Intelligence (AmI) concept.

In the following sections we briefly describe our methodology (section 2), we present CBM content (section 3), then we discuss the formalism features and present ORCHESTRA concepts (section 4). After that we discuss pattern approach and give several patterns (section 5). Finally, an illustrative example (section 6), conclusions and perspectives are finishing the paper.

2 Our Approach: CoCSys Methodology

We are studying design of CSCW systems and we propose an approach and a process, called CoCSys (Collaborative Capillary System) engineering process. Main reason for this more comprehensive process is related to the necessity to allow the evolution of this kind of system during its use in relation with the users’ skills, expertise, and the evolution of their perception and the mastery of the system. Our approach is based on Model-Based approach [17], which is characterized by a different way of development: “Rather than programming an interface using a toolkit library, developers would write a specification of the interface in a specialized, high-level specification language. This specification would be automatically translated into an executable program, or interpreted at run-time to generate the appropriate interface.” This approach is used in HCI for several years and become more generally used in other development application fields. OMG adapted a similar approach as new paradigm of development which is called MDA Model-driven architecture [14]. Other acronyms describing similar ways are MDE (Model-Driven Engineering) or MDD (Model Driven Development). In each case specification at concrete, abstract or meta level is privileged before studying the way to produce an executable code. The production is done more or less automatically by transformation or translation of these models. The objective of our approach is to adapt this trend to CSCW. We are proposing a framework for design, implementation and evolution of CCS. As described deeply in [5, 7] this approach is based on 3 main parts: 1/Scenarios Collection, 2/Cooperative Behaviour Model (CBM), and 3/Collaborative Architecture; and 3 transformation phases: I/CBM Model Construction, II/CBM Projection on the Collaborative Architecture and III/Evolution.

3 Scenarios and Cooperative Behaviour Model

We consider that a scenario allows to final users and designers to meet them and discuss together about functionalities of the system to be developed. A scenario describes repetitive activity that should activate an adaptation mechanism which will be recorded and reused. For us the scenarios are short stories describing precise working situations which occur for different actors. This analytical perception of working situations seems be possible to catch and express observers or actors needs. We are asking to give as precise description as possible, i.e. to indicate, if possible, all actors evolving, artefacts used, activities executed and contexts characterising them (devices used, geographical location, temporal situation …). We collect these scenarios for different collaborative situations. In this way we can consider that this formulation of scenarios is possible, meaningful and useful. If scenarios are short limited stories, expressed mainly by different actors, behaviour model objective is to discover overall organization of the cooperative system in which main elements are actors, artefacts, tasks, processes and contexts. The designers are in charge to study different scenarios and to construct gradually the Cooperative Behaviour Model (CBM). In the model we find comprehensive collections of actors, artefacts, activities and contexts and also all relations which allow materializing all necessary elements for each activity. Different processes are also explained carrying out dependencies between tasks and their temporal and organizational constraints. This comprehensive model is able to manage the cooperative system behaviour and will be used during the implementation process i.e. projection of this model on a particular hardware, network and software architectures. Main elements of the CBM model are:

- An **actor**, as instantiation of one or several roles, a role is a basic element of human behavior in the system, which can be qualified as Acting (A), Observing (O) or Editing (E) i.e. observing and acting.
- An **activity**, describing an identified work which a role can do, this activity can be also A, O or E, i.e. acting, observing or editing activity.
- A **process** expressed as a network composed of process states (PS) and process transitions, which can also be qualified by A, O or E.
- An **artefact** can be either a tool or an object. The tool is an instrument used in the task; the object is either input, support or output of the task, qualified by A, O or E.
- A context is a collection of three aspects giving platform, situation (often logical, physical or geographical location) and user preferences characterising the context. We take into account several platform examples and elements: laptop, PDA, cellular phone, and also active environmental object (active RFID tag), passive environmental object (passive tag), …

In the CBM model all these elements are expressed and interconnected. We can take as example a user's role, which is identified by a name, a type, its participation in different actors, the activities which can be done, the process states and transitions in which their can occur, the artefacts (tools and objects) manipulated and the contexts (platform, situations and user preferences) which applies the role. These interrelations are also needed for other elements of the model. They are explicitly or implicitly described and can change during the system life expressing its adaptation and

evolution. List of activities is one of the main components of CBM. This list is obtained from the task tree which can be expressed by CTT [15], an interesting task formalism, and its environment (CTTE) proposed by Paterno. Its extension for cooperative activities [13] aims to express cooperative situations. In CTT, collaboration is expressed by individual task trees and by a collaborative task tree. That is interesting to express tasks, but is insufficient for the more comprehensive view of collaboration, that we need. We consider that tree view of tasks is interesting during the task design phase. However, during the activities organization (definition of effective collaborations), mainly effective activities (leaves of the task tree) are important and their individual or collaborative scope is essential, in relation with effective actors, objects, tools, process states and transitions and contexts. To express in a more comprehensive way we propose a new formalism called Orchestra [5].

4 ORCHESTRA

The objective of Orchestra is to propose a more comprehensive formalism which is able to express together all main aspects of the CBM. ORCHESTRA adapts musical score notation [18] to our problem of CBM description. For us, the 5 lines of a **staff** are expressing 5 main aspects of the CBM (Fig. 1), which are: user's role, activity concerned, process state or transition, artefacts involved in the activity and the context. These aspects are expressed on each of their respective line by situating one or several "**notes**" containing their names. Each note can receive a **stem** which indicates the participation of the element (acting, observing or editing). We distinguish main actor (double arrow) and secondary actor (simple arrow) as well as active role and passive role:

- ✡ Active role
- ◎ Passive role

A bar line indicates the separation between independent **cooperation episodes**. To express **repetition** of an episode we propose four options: an explicit number of repetitions (**n**), an undetermined number of iterations (**+/***), a contextual end (**logical condition**), a time dependent end of iteration (**relative or absolute time limit**).
Each cooperation episode expresses a state or a transition in the cooperation process description network. For each cooperation episode, **sequential ordering** from left to right is implicit temporal option, another order, must be expressed explicitly either by a **jump** from current period to another one which is named, or by a "**procedure call**" jump to a named episode then the back to the previous one.

By different types of parenthesis, we indicate explicit relations between participating notes. These parentheses are used to express different situations:

(…) alternatives,
{…} mandatory participation,
[….] optional participation.

Different key signatures are expressing collaboration properties like synchronous or asynchronous collaborations, collaboration modes and styles of coordination (computational 💻 or social ☺, implicit ● or explicit –):

@ - **Asynchronous** with infinite answer delay
@@ - Asynchronous with limited answer delay corresponding to "**on call**" participation
& - Synchronous "**in-meeting**" cooperation
&& - Synchronous "**in-depth**" cooperation

In synchronous collaboration two different participations must be distinguished:

- **instantaneous**, short term collaboration, called also implicit and expressed by ● i.e. vote activity,
- **long term** participation, long term collaboration, called also explicit and expressed by ▬ i.e. sketching activity.

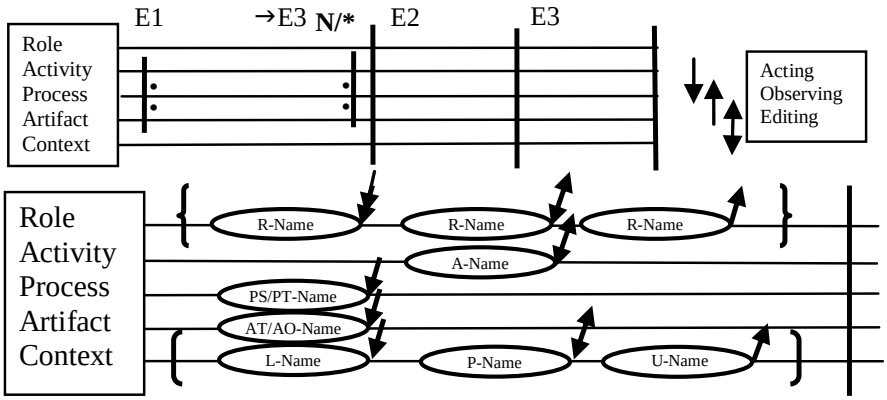


Fig. 1. ORCHESTRA main concepts

In the first case (vote activity) an implicit collaboration is appropriate (short exclusive access to the shared space), in the second case (sketching) explicit participation must be asked and allowed (long-term access to the shared space) either by **social coordination** (😐), i.e. one of human actors is in charge of this coordination or a **computational** (💻) one i.e. the computer fulfil it. We express graphically instantaneous collaboration by a **dot** over concerned chords and for long term collaboration we use a horizontal line ▬ and a symbol expressing social or computational coordination (😐, 💻) i.e. coordination made by one of the actors or by interaction (asking for, receiving and returning exclusive access right to shared space).

Another important notion in CSCW is **awareness**. Its objective is to allow to different actors to know (or not) what has been done by an actor. It is important to decide statically (by the designer) or dynamically by the actor himself the scope of information propagation to other actors. For static way we propose to express awareness in ORCHESTRA formalism. Special marks are proposed:

- for **no awareness**,
- for **partial awareness** (for specific actors),
- for **overall awareness** (for all actors).

To explain more deeply ORCHESTRA formalism, we give in [5] its metamodel which contents ORCHESTRA and CBM metamodels.

5 Patterns

As initially expressed by Christopher Alexander [1]: “A pattern is a careful description of a perennial solution to a recurring problem within a building context, describing one of the configurations which brings life to a building.”

In software engineering a Design Pattern describes a family of solutions for a given Software-Design problem. The Pattern is not the solution itself, but a solution framework. The final goal of Design Patterns is the reusability of Software Design knowledge. Patterns can be used for different reasons, as: to improve team communication, to document and facilitate the state-of-the-art and to reflect main concepts. Patterns can also help to understand, to clarify and document design decisions. They can help to avoid design drift and also can improve code structure and code quality. For these reasons, patterns can be useful everywhere (in process, product and activity), as reusable problem-context-solution descriptions. Methodological, functional, process, analysis, scenario, testing and evaluation patterns are proposed and useful, as well as design, HCI, UI patterns.

In our case, we propose patterns for ORCHESTRA which objective is to express in a reusable manner main cooperation situations. Our approach of patterns is in relation with Alexander [1], Gamma [11], Borchers [3] and Seffah [12], we adopt a more comprehensive and generic definition: A pattern is a collection of elements and their relationships. They can be repetitively reached or used in analysis, design, development and use (of cooperative systems): **Pattern = Problem + Context + (potential) Solution(s)**.

It seems important to highlight the convergence of interest between different patterns users. In HCI design and groupware design, patterns are useful for the designer (professional) as expression of best practices, standardization and usability; they are also useful for the final user for standardization (same thing is done in the same manner in different situations), learnability and usability.

In figure 2 we are giving an open-ended list of ORCHESTRA patterns. They are either finalized chords with appropriate annotations, or and more usually incomplete configurations which could be completed during the instantiation process. Chords are mainly generic, i.e. role, activity, process, artefact (tool or object) and context can be chosen from corresponding concrete application field instances.

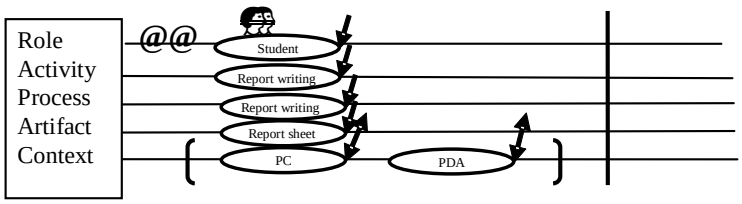
ORCHESTRA pattern is a schema with one or several chords constituting cooperation episode(s) organized temporally and associated to a particular configuration of complementary annotations expressing **nature of cooperation** (Synchronous or Asynchronous), **level of cooperation** (asynchronous with infinite delay, on call, in meeting or in-depth cooperation), **coordination style** (social or computational), **nature of coordination** (implicit or explicit) and **awareness** (overall, partial or no awareness). To exemplify this approach we are able to present six important cooperation patterns which are the following:

- **Intervention appointment:** Synchronous or asynchronous, on-call or in-meeting cooperation with computational implicit coordination and no awareness.
- **Consultation – vote:** Synchronous, in-meeting cooperation with computational and implicit coordination and either overall awareness or no awareness.
- **Presentation:** Synchronous and in-meeting cooperation with social and explicit coordination with overall awareness.
- **In-depth work:** Synchronous, in-depth cooperation with computational explicit coordination and partial awareness.
- **Questions / Answers:** Synchronous, in-meeting activity with social or computational explicit coordination.
- **Validation:** Asynchronous, on-call cooperation with implicit coordination and no awareness.

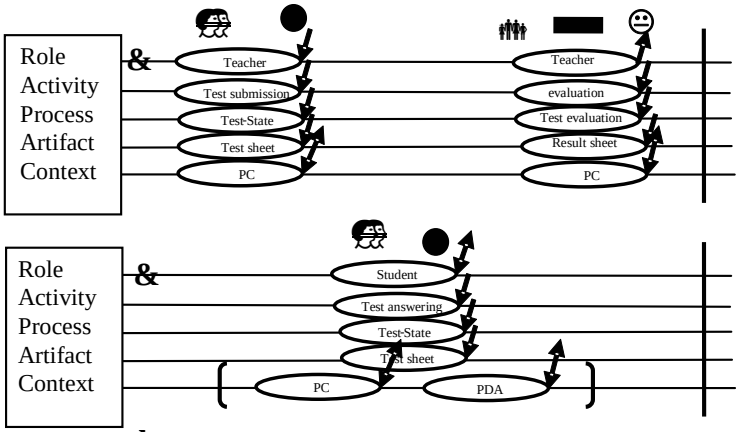
Pattern	S/As	Coop	Coord	Exp/Imp	Aware	Coop. configurations
Intervention appointment	S/As	&/@@	😐	●		
Consultation Vote	S	&	💻	●	/	
Presentation	S	&	😐	▬		
In-depth work	S	&&	💻	▬		
Questions / Answers	S	&	😐 / 💻	▬	/	
Validation	A	@@		●		

Fig. 2. Characterisations of several ORCHESTRA patterns

We give on the figure 3a ORCHESTRA description of report writing activity which is an instantiation of validation pattern and on figure 3b a description of test activity which is an instantiation of vote pattern. Names in inside of notes are formal; they will receive final names during instantiation of patterns.



a - Individual activity "Report writing"



b - Collaborative test activity preparation, execution and treatment

Fig. 3. Two ORCHESTRA patterns instantiations

6 Case study: Heating equipment maintenance activities

To explain ORCHESTRA formalism use we are expressing with it heating equipment maintenance activities (Fig. 4) with six actors: client, secretary, technician, supervisor, expert and clerk. Main scenarios of maintenance process are the following:

- A client (secondary actor), observing a problem with his heating equipment, phones to the repair company to ask intervention. The secretary (secondary actor) asks him his profile (address, equipment…) and finds him in the database. He organizes an appointment with a technician. **State: Appointment, Actors: Client, Secretary, Properties: &**
- In the morning, before leaving the company, the technician (main actor) loads on his PDA necessary information for his round with appropriate information (clients and their addresses, nature of intervention …). **State: Init, Actor: technician, Properties: @**
- At client house, the technician works on maintenance process, he can study history file of the supplies and blueprints, to elaborate a diagnosis using appropriate tools, and repair, or ask for spare parts. **State: Work, Actors: Client, Technician, Properties: &**
- In a situation of impossibility to establish a diagnosis alone, he can contact his manager (secondary actor) to ask him some helps and to exchange some information. He can also contact, in a synchronous manner the heating manufacturer expert (secondary actor) to study the situation with him. **State: Coop, Actors: Technician, Manager, Expert, Properties: &&**
- At the end of his round, the technician, back to the company, updates history file of visited equipments and gives his intervention statement. **State: End, Actor: Technician, Properties: @**

- Next day the clerk (secondary actor) produces the financial balance and statement of accounts and either sends the bill to the client or integrates it in the client record. **State: FB (Financial Balance), Actor: clerk, Properties: @**

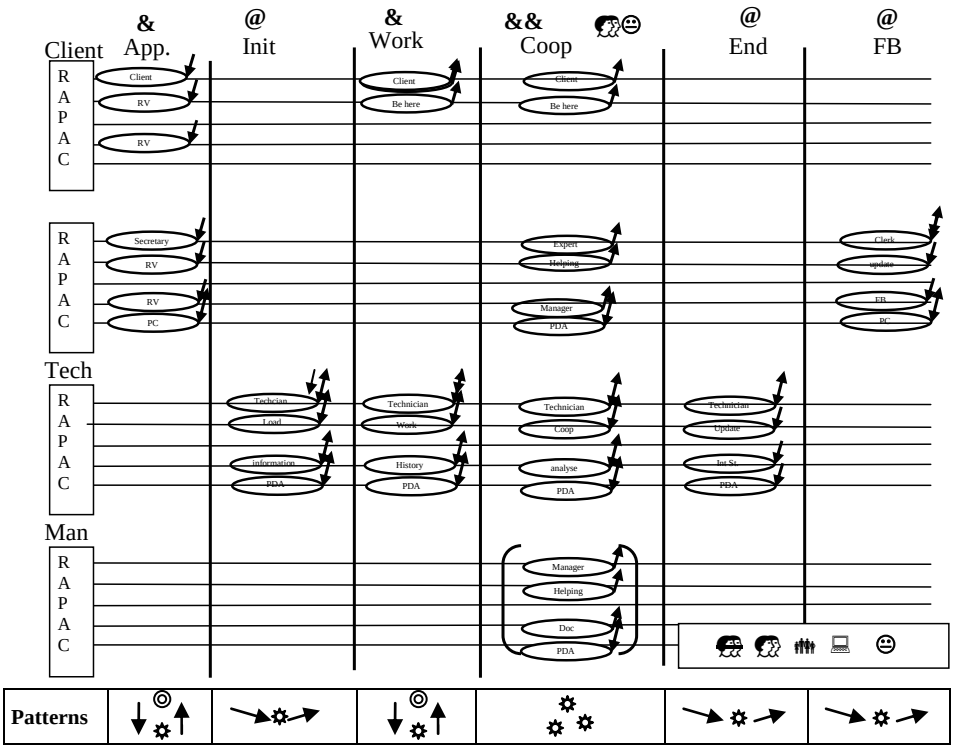


Fig. 4. Different ORCHESTRA description of heating maintenance activities example

In figure 4 we show ORCHESTRA modelling for this case study and associated patterns. Individual activities are expressed on one staff. For collaborative activity, several staffs are needed, each for a role. In this way we describe on the same sheet, the participation of each actor to this collaborative episode and we facilitate its understanding.

Conclusion

In this paper, we outlined a new formalism called ORCHESTRA, which objective is to provide a graphical expression of Cooperative Behavior Model. CBM, elaborated from a collection of scenarios, as a reference for the transformation process allowing different implementations. As it is important to associate different actors to this constructive process, we propose a formalism which could be used during initial discussions as well as during the implementation and adaptation process. We presented a set of reusable patterns which are useful to accelerate and do design

process more powerful. We propose to use them in a pattern oriented walkthrough, in which patterns are considered as best practices, as a collection constituting an inspiration sourcebook and as a use guide. We presented ORCHESTRA use on a case study. Of course ORCHESTRA explains a global view of cooperation. An in-depth view is necessary to describe completely the content of “notes” with the help of an editor.

ORCHESTRA has been tested in several case studies and we may continue to upgrade it by new concepts as result of these tests. The connection with mixed reality has not been described in this paper, even if we are currently working on it.

References

1. Alexander, C., Ishikawa, S., Silverstein, Jacobson, M., Fikdhl-King, I., Angel, S.: A Pattern Language: Towns; Building, Constructions. Oxford University Press, New York (1977) 1216 p.
2. Andriessen, J.H.E.: Working with Groupware: Understanding and Evaluating Collaboration Technology. Springer, CSCW Series (2003) 206 p.
3. Borchers, J.O: A Pattern Approach to Interaction Design. In ACM Press John Wiley & Sons (2000) 369 – 378
4. Chalon, R., David, B.: IRVO: an Interaction Model for designing Collaborative Mixed Reality Systems, HCI International 2005, Las Vegas, USA, 22-27 July (2005)
5. David, B., Chalon, R., Delotte, O., Masserey, G., Imbert, M.: ORCHESTRA: formalism to express mobile cooperative applications. In Book Series Lecture Notes in Computer Science, Vol. 4154, Springer, (2006) 163-178
6. David, B., Chalon, R., Vaisman, G., Delotte, O.: Capillary CSCW. In Proceedings of HCI International, Crète (2003)
7. David, B., Delotte, O., Chalon, R.: Model-Driven Engineering of Cooperative Systems. In proceedings of HCI International 2005, Las Vegas, USA, 22-27 July (2005)
8. David, B. : IHM pour les collecticiels. In Réseaux et Systèmes Répartis, Hermès, Paris, vol. 13 (2001) 169–206
9. Ellis, C., Gibbs, S.J., Rein, G.L.: Groupware: some issues and experiences. In Communications of the ACM, vol. 34, n° 1, (1991) 38–58
10. Ellis, C., Wainer, J.: A conceptual model of Groupware, In Proceedings of CSCW'94, ACM Press, (1994) 79–88
11. Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns, Elements of reusable Object-Oriented Software, Addison-Wesley Publishing Company (1995).
12. Javahery, H., Seffah, A., Engelberg, D., Sinnig, D.: Migrating User Interfaces between Platforms Using HCI Patterns. In: Seffah A.; Javahery H.; (Eds): Multiple User Interfaces: Multiple-Devices, Cross-Platform and Context-Awareness. Wiley (2003) 241-259
13. Mori, G., Paternò, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. In IEEE Transactions on SE, vol. 28, n. 9 (2002)
14. Object Management Group, http://www.omg.org/mda/
15. Paternò, F.: Model-Based Design and Evaluation of Interactive Applications. Applied Computing Series, Springer -Verlag (2000)
16. Rosson, M.B., Caroll, J.M.: Usability engineering scenario-based development of human-computer interaction. Morgan Kaufmannn (2002)
17. Szekely, P.: Retrospective and Challenges for Model-Based Interface Development. In: Vanderdonckt, J. (eds): CADUI'96, 5-7 June 1996, Namur (1996)
18. Stewart, D., The Musician’s Guide to Reading and Writing Music. Backbeat (1999) 117 p.