LIVE PATCHING COLLABORATIF - VERS UNE MEDIATION INCLUSIVE AVEC L'INFORMATIQUE MUSICALE

João Svidzinski, Messina Marcello

LIVE PATCHING COLLABORATIF - VERS UNE MEDIATION INCLUSIVE AVEC L'INFORMATIQUE MUSICALE

João Svidzinski
Université Paris 8
Laboratoire Musidanse/CICM
UFPB – Federal University of Paraiba
svidzinski@gmail.com

Marcello Messina

UFPB – Federal University of Paraiba
João Pessoa, Brésil
marcellomessina@mail.ru

RÉSUMÉ

Nous présentons dans cet article un projet *work in progress* avec *live patching* au sein d'une collaboration avec le Conservatoire de Saint-Denis. Après avoir utilisé le logiciel Kiwi pour cette pratique, nous migrons vers Pure Data et Faust en poursuivant notre conception du *live patching,* notamment l'approche « commencer du zéro ». C'est-à-dire, construire collaborativement et en temps réel un patch sans aucun a priori, à partir d'un document vide. Cette pratique a été appliquée au sein d'un atelier avec le Conservatoire de Saint-Denis, aboutissant sur une représentation publique le 29 mars 2021 à la Maison des Sciences de l'Homme Paris Nord. Pour cette restitution, nous avons choisi de présenter une séance construite à partir des nos ateliers avec *live patching*. Plutôt que de coder en temps réel, les participants ont joué avec un patch déjà créé, mais pouvant interagir avec des diapositives OSC et avec leurs instruments musicaux.

1. INTRODUCTION

Le Conservatoire de Saint-Denis collabore avec nous depuis 2016 au sein de nos projets de recherche-création hébergés à la Maison des Sciences de l'Homme Paris Nord – MSH PN. Le projet phare de cette collaboration est le concert annuel « Musique Mixte » au sein duquel des pièces mixtes composées par les étudiants de l'université Paris 8 sont interprétées par les élèves, majoritairement adolescents, du Conservatoire de Saint-Denis. À partir de 2020, ce projet a dû subir des modifications, notamment pour l'adapter à une nouvelle réalité sociale liée à la Covid-19. Dans un premier temps, une nouvelle activité a été proposée en 2020. Les enseignants du Conservatoire ont été formés en informatique musicale afin qu'ils puissent eux-mêmes enseigner cette discipline [2]. Pour cette formation le logiciel Kiwi a été utilisé avec Faust.

Parallèlement, depuis début 2021, le projet « Organisations musicales symbiotiques : une révision de la notion de concert de recherche-création faisant appel à l'informatique musicale » a lieu à la MSH PN. Son objectif est de revisiter la pratique actuelle musicale notamment par le biais de la musique en réseau. L'une des activités encadrées par ce projet consiste à réaliser des séances de *live patching* avec un groupe d'élève du Conservatoire. Cette pratique a été choisie, car nous organisons depuis 2018 des séances de *patching* collaboratif, spécialement en rapport avec le développement du logiciel Kiwi. De cette démarche, a découlé notre recherche de post-doctorat « *Live patching com kiwi e Faust: uma proposição artística e pedagógica para a prática musical universitária* » à l'Université Fédéral da Paraíba, à João Pessoa (Brésil). Cette recherche est menée en collaboration avec le projet « Live/Acc/Patch » basé en deux universités brésiliennes, Université Fédéral do Acre et Université Fédéral da Paraíba.

En guise de préparation pour les séances de *live patching* avec le Conservatoire de Saint-Denis, nous avons organisé une séance interactionnelle de *live patching* avec Kiwi et Faust dans le cadre de *International Faust Conférence* (IFC-20) [13]. À l'issue de cette expérience, nous avons constaté que, en l'état actuel, le logiciel Kiwi n'est plus optimisé pour une pratique collaborative. Comme son développement n'est plus assuré, dû à la fin du projet ANR MUSICOLL en 2018, ses fonctionnalités ne sont plus à jour. Pour la suite du projet, nous avons décidé de continuer nos activités avec Pure Data et Faust, tout en conservant le principe le plus cher du *live patching,* l'approche compositionnelle « commencer du zéro ».

Dans cet article, nous allons décrire brièvement le *live patching* pour ensuite présenter notre conception de cette pratique. Cela correspond essentiellement à l'approche « commencer du zéro ». Finalement, nous allons présenter notre activité au sein de la collaboration avec le Conservatoire de Saint-Denis.

2. LIVE-PATCHING

Le *live patching* dérive de *live coding*. Cela est défini par Thor Magnusson comme « une forme de représentation musicale qui implique une composition musicale en temps réel à travers le code textuel » [6]. Le *live coding* est devenu une pratique créative, ainsi qu'un domaine de recherche au cours des dernières années. Collins situe le *live coding* comme une nouvelle pratique musicale fondée sur le risque, l'imprévisibilité et, par conséquent, le grand potentiel de génération d'événements inattendus [3]. Magnusson, quant à lui, met en lumière le

passage de la partition traditionnelle à l'algorithme, introduit par le *live coding*, comme une nouvelle façon de créer du sens musical [6].

Le duo « Slub » formé dans les années 2000 par Alex McLean et Adrian (Ade) Ward est un exemple pionnier dans ce domaine [3]. Ils développent leurs propres logiciels audios avec des langages comme Perl et REALbasic, et les exécutent en direct de manière synchrone via un serveur TCP / IP. Pendant leur performance de *live coding,* les auteurs partagent leur écran montrant la séquence de lignes de commande *bash shell*. Actuellement, Alex McLean organise des sessions de *live coding*, souvent diffusées via YouTube[1]. Le compositeur utilise le langage Tidal pour générer des motifs rythmiques typiques de la musique techno.

Le *live patching* a été introduit par les projets *blind date* et *rec.wie.m* de l'association Pd ~ Graz [11]. Il s'agit d'expériences collectives de *live patching,* en privilégiant les environnements de programmation graphique. Cette fonctionnalité facilite la formulation de stratégies visant à rendre plus facile la compréhension de la syntaxe modulaire et ainsi, accélérer l'apprentissage des utilisateurs sans aucune expérience de ce type de langage.

2.1. De Kiwi à Pd avec Faust

Dans ces mêmes perspectives des logiciels graphiques comme Max et Pure Data, le logiciel Kiwi ont été développés dans le cadre du projet ANR MUSICOLLL (2016-2018). L'une de ses qualités principales est le *patching* collaboratif. C'est-à-dire, ce logiciel permet à plusieurs utilisateurs de composer ensemble en temps réel un unique processus musical sur un patch hébergé en ligne [9][10]. Ainsi, les utilisateurs n'ont pas besoin d'être géographiquement réunis pour faire de la musique ensemble. Cette pratique est fortement envisageable dans un contexte de distanciation sociale. Jusqu'à présent, Kiwi a été utilisé plutôt dans des cadres pédagogiques [4]. En effet, il s'avère puissant pour l'enseignement de la musique mixte à distance. Cependant, une nouvelle manière de faire de la musique est en train d'émerger. Depuis 2019, ce logiciel est utilisé pour la pratique du *live patching* intercontinental au sein d'une collaboration entre l'université Paris 8 et du projet de recherche Live/Acc/Patch de deux universités brésiliennes : Universidade Federal do Acre et Universidade Federal da Paraíba [7][8]. En 2018, plusieurs séances ont été organisées entre ces institutions. Le principe était simple : à partir d'un document vide, commencer à créer un patch ensemble. Comme Kiwi partage seulement la représentation graphique ainsi que quelques signaux de contrôle, le rendu sonore était unique et parfois assez différent pour chaque participant. De ces expériences, plusieurs questions pratiques ont découlé, notamment en rapport avec la confrontation de différentes « cultures musicales », avec des savoir-faire distincts, autour d'une pratique partagée.

Afin de continuer cette recherche, en décembre 2020, lors de la IFC-20 - *International Faust Conference 2020* (IFC-20), nous avons organisé un premier workshop « *Live patching with Kiwi and Faust : an intercontinental experience* » [13]. Nous avons attesté que le logiciel Kiwi n'était plus adapté pour la pratique du *live patching* intercontinental. Avec la fin du projet ANR MUSICOLLL en 2018, son support n'était plus assuré.

Le développement du logiciel Kiwi mérite une continuation. Pour l'heure, nous avons décidé de poursuivre les perspectives créatives lancées par Kiwi avec d'autres outils, comme Pd. Les fonctionnalités collaboratives ne sont pas assurées dans les fonctions basiques de Pd[2]. Ainsi, les principes fondateurs du *live patching* ont été détournés. À l'origine, cette modalité musicale prévoit que plusieurs auteurs composent en temps réel une musique collaborative. Le patch est normalement projeté et le résultat sonore est partagé entre les participants. Pour nos séances avec Pd et Faust, nous avons quand même retenu la caractéristique principale de cette pratique observée lors de nos activités précédentes : la création d'un patch collaborativement « du zéro ».

2.2. L'approche compositionnelle « commencer du zéro »

Pour nos premières expériences de *live patching* intercontinental en 2018, aucune règle n'a été déterminée au préalable. L'objectif était de construire un patch ensemble à partir d'un document vide. Pour la première séance, deux groupes d'étudiants y ont participé. Un premier formé par des étudiants de Licence 2 en musique à l'université Paris 8 de la filière « Composition assistée par ordinateur », spécialement les étudiants du cours « Programmation avec Kiwi, Max et Pd 1 ». Le deuxième groupe, constitué d'étudiants et de jeunes chercheurs de deux universités brésiliennes. Ceux-ci avaient une formation musicale large sans spécialité en informatique musicale. Lors de la première expérience, le premier groupe a commencé le patch par le développement des modules pré-codés de synthèse FM (sujet qui venait d'être abordé dans la discipline), alors que le deuxième groupe commençaient par l'agencement des objets isolés, comme des oscillateurs et générateurs de bruit. Cette différence n'est pas anodine. Or, la notion de « commencer du zéro » n'était pas la même entre les deux groupes. À partir de la deuxième séance, il a été convenu que tous les participants commenceraient par l'agencement des objets individuels. Cette première expérience montre déjà que le principe fondateur du *live patching* est assez relatif. Ce constat est encore plus flagrant lors de l'intégration du langage Faust. Cela a été observé lors du workshop avec Faust et Kiwi dans le IFC-

[1] https://www.youtube.com/user/yaxu (lien vérifié le 5 mai 2021).

[2] Il existe quelques tentatives d'intégration de ces fonctionnalités, comme le Pd-Net. Mais son installation et configuration pose de problèmes pour les utilisateurs novices. De plus, la version *stand-alone* n'est aussi modulable que Kiwi.

20. En même temps que certains participants développaient des codes Faust « du zéro », certains modifiaient des modules dans le patch Kiwi. Les deux approches sont cohérentes avec le principe du « commencer du zéro ».

Le degré du *live* est donc relatif et cela peut être un moteur fondateur pour nos prochaines activités. Pour les ateliers de *live patching* avec le Conservatoire de Saint-Denis, nous avons décidé de concevoir le *live* en trois étapes différentes. Tout d'abord, par le développement des outils en Faust. Cela peut se faire collectivement par des étudiants de l'université Paris 8, ainsi que par l'appropriation des codes déjà développés. Ensuite, ces modules sont choisis, « joués », et modifiés en temps réel par des élèves au cours de trois séances. Finalement, le patch « final », développé au cours de trois séances, sera interprété lors d'une restitution publique.

3. MEDIATION INCLUSIVE

Une deuxième caractéristique du logiciel Kiwi retenue pour les ateliers de 2021 est l'absence de hiérarchie entre les participants. Dans Kiwi, tous les participants conservent les mêmes droits sans restriction. Ainsi, les opérations sur chaque patch ne laissent pas de traces génétiques, c'est-à-dire qu'il est impossible de savoir qui a créé un objet spécifique ou qui a créé un commentaire dans le patch. De cette manière, les potentielles barrières hiérarchiques (par exemple, enseignant / étudiant ou groupe X vs groupe Y) sont totalement évitées. Cette caractéristique est souvent comprise comme une défaillance logicielle. Du point de vue du développement informatique, il est souhaitable que l'auteur d'un document partagé puisse choisir ou autoriser la collaboration d'un auteur supplémentaire. Cela n'est pas le cas dans Kiwi.

Notre rôle dans cette activité est essentiellement d'être médiateur, c'est-à-dire d'un intermédiaire entre les élèves et la pratique compositionnelle avec l'informatique musicale. Plutôt que de les encadrer dans un programme figé, nous avons soutenu leurs interventions pour le développement d'une structure musicale. Ainsi, chaque élément de notre représentation a été soigneusement choisi par tous les participants, y compris les matériaux sonores, les traitements numériques et les modes de jeux instrumentaux. Ainsi, la composition est fondée sur les échanges.

En seulement trois séances, nous avons préparé une représentation digne d'une restitution publique réalisée par des élèves instrumentistes non-initiés à l'informatique musicale. Tout cela en appliquant les principes du *live patching*, spécialement notre notion du « commencer du zéro ». Il s'agit d'une entreprise audacieuse, mais dans l'esprit de la recherche-création aussi chère au sein de nos activités.

3.1. Séances collaboratives

Les ateliers au Conservatoire de Saint-Denis ont impliqué la participation d'un groupe de trois élèves instrumentistes adolescents, deux saxophonistes et un violoniste. Ils n'avaient aucune pratique d'informatique musicale. Pour les initier à la pratique du *live patching,* chacune des trois séances a été divisée en deux parties. D'abord une phase d'apprentissage de la syntaxe de Pd et ensuite, un temps de pratique, où ils pouvaient tester les patchs à la fois en tant qu'opérateur du patch, et qu'instrumentistes. Malgré la volonté des élèves, nous avons rapidement remarqué qu'avec seulement trois séances il serait impossible de préparer une représentation publique de *live patching.* De plus, les élèves n'osent beaucoup participer. Or, le *live patching* attend des participants qu'ils soient actifs et dynamiques. Nous avons donc immédiatement changé de stratégie. Deux idées déjà, pré-réalisées, leur ont été présentées. D'abord un exemple de synthèse avec Faust et un deuxième un traitement temps réel, aussi en Faust, avec des lignes retards, afin de créer des patterns rythmiques.

Concernant la synthèse, le point de départ à été un objet Faust compilé en Pd. Il s'agit d'une variation du *bubble.dsp,* l'un des codes exemples de la distribution du langage[3]. Pour notre version, une fonction *button("play")* a été ajoutée comme argument *trig* de *bubble(f0,trig)*. Cela permettra son déclanchement dans le patch Pd. Cet exemple a été choisi, car il est relativement simple, avec des paramètres compréhensibles par les élèves (fréquence, amplitude). Il génère aussi une réponse « causal », c'est-à-dire, à chaque fois qu'un participant appuie sur le bouton « play », un son de « goutte d'eau » est généré[4].

Pour les premières expériences, les élèves contrôlaient directement le patch à tour de rôle sur un même ordinateur alors que les autres expérimentaient avec leurs instruments (figure 1).

Figure 1. Première séance de travail : un étudiant expérimente l'électronique alors que deux autres interagissent avec leurs instruments.

[3] https://github.com/grame-cncm/faust/blob/master-dev/examples/gameaudio/bubble.dsp (lien vérifié le 5 mai 2021).

[4] L'ensemble de materials, codes et patchs, est accessible dans https://github.com/CICM-research-composition/livepatching (lien vérifié le 5 mai 2021).

L'élève qui manipulait le patch pouvait changer la fréquence de la « goutte d'eau », ainsi que sa périodicité. Pour ce dernier contrôle, un objet Pd *metro* avec un *bang* était connecté au trigger du code Faust ci-dessus. Cela leur permettait de modifier la fréquence temporelle ou de le déclencher manuellement. Pendant qu'un élève testait ce système les autres interagissaient avec leurs instruments, tout en prenant en compte la nature de la morphologie sonore générée dans leurs jeux instrumentaux.

Quant à la deuxième idée, concernant le traitement temps réel, nous avons repris un code réalisé par un étudiant de l'université Paris 8 et déjà utilisé lors d'un autre atelier au même Conservatoire [1]. Notre hypothèse était fondée sur le fait que, comme cet objet offre des contrôles proches de la musique traditionnelle, comme le tempo et les notations rythmiques traditionnelles, les élèves seraient confortables avec son utilisation. Cependant, à notre surprise, ils étaient plus sensibles au son lui-même qu'aux contrôles musicaux. Ils voulaient plutôt créer de nouveaux sons pour faire un parallèle avec la musique qu'ils connaissaient déjà. Ainsi, nous avons abandonné cet objet pour une version simplifiée de *l'harmonizer* avec le principe de réinjection matricielle [12]. Son grand potentiel générateur des morphologies sonores a motivé notre choix.

3.2. Restitution publique

La dernière des trois séances au Conservatoire a été consacrée à la structuration de la restitution publique. Afin que tous les participants puissent interagir de manière synchrone, nous avons créé une application OSC[5] accessible par un téléphone portable et connectée à un réseau local (figure 2). Dans cette application, chaque participant pouvait manipuler quatre groupes de GUI[6], chacun correspondant à un haut-parleur. Chaque groupe permet de contrôler la fréquence des « goutte d'eau » et sa périodicité. Un bouton permet d'activer le *metro* et un deuxième permet de déclencher manuellement une « goutte d'eau ».

Figure 2. Application OSC accessible par un téléphone portable.

Les applications OSC était connectées au même réseau du patch Pd avec les objets Faust. Pour que Pd communique avec OSC, nous avons eu recours à la bibliothèque *mrpeach*[7].

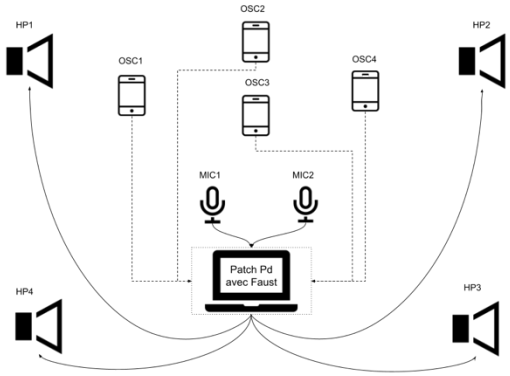

Figure 3. Schéma technique de la restitution.

L'installation technique de la restitution est alors illustrée dans le schéma ci-dessus (figure 3). Chaque participant, avec son propre téléphone portable interagit via l'ordinateur avec Pd. Deux micros permettent aux élèves de jouer avec leurs instruments. Le rendu sonore sort en quadriphonie.

Figure 4. Première partie de la restitution (les quatre participants interagissent avec leurs portables).

Un script a été préparé en précisant les étapes du déroulement pour la restitution. D'abord, les étudiants entrent en scène avec leurs portables et discrètement commencent à jouer avec les applications OSC (figure 4). Il s'agit d'une métaphore de la réalité actuelle. Avec la distanciation sociale, les personnes doivent interagir de façon distribuée, fréquemment par l'intermédiaire de leurs téléphones portables. Cela n'empêche pas de faire de la musique ensemble. Au-delà d'une caricature sociale, nous avons remarqué que le portable est un excellent outil pour ce genre d'activité. Il permet à la fois de servir d'intermédiaire entre le quotidien des élèves adolescents avec une vraie pratique musicale. De plus,

[5] Réalisé avec Open Stage Control https://openstagecontrol.ammd.net (lien vérifié le 5 mai 2021).

[6] Interface graphique (GUI *: graphical user interface*).

[7] https://puredata.info/downloads/mrpeach (lien vérifié le 5 mai 2021).

grâce à son caractère anonyme (le public ne peut pas affirmer avec certitude qui est en train de générer chaque son), les élèves sont souvent à l'aise pour se libérer du trac et prendre des risques musicaux.

Figure 5. Suite de la restitution (les élèves jouent avec l'électronique alors que les autres continuent à interagir avec leurs portables).

Pour la suite de la restitution, les élèves lâchent leurs portables et jouent à tour de rôle leurs instruments (figure 5). Le script a été conçu d'une manière qui permet à chaque élève de faire des « solos » et que tous les trois jouent ensemble. Même si cette partie peut ressembler à une improvisation. Les élèves doivent prendre en considération la globalité de l'environnement sonore, à savoir, le son généré par l'électronique ainsi que les jeux instrumentaux des collègues lorsqu'ils jouent ensemble. Ainsi, les participants font preuve d'importantes notions musicales, comme l'écoute, l'improvisation et la création.

4. CONCLUSION

La restitution publique de l'atelier a eu lieu le 29 mars 2021 dans l'auditorium de la MSH PN. La captation vidéo sera diffusée dans le cadre du festival DéMO du Conservatoire de Saint-Denis et sera disponible en ligne[8].

Cette activité fait suite aux recherches initiées avec le projet ANR MUSICOLL qui a encadré le développement du logiciel Kiwi. À partir de l'apprentissage antérieur avec cet outil, nous avons développé un savoir-faire sur le *patching* collaboratif. Le retour à l'usage de Pd a entrainé quelques modifications dans notre pratique. Contrairement au *live patching* avec Kiwi, les interactions étaient asynchrones, c'est-à-dire, réalisées au même lieu en des moments différents. Cela implique que la notion du *live* a été distordue. Même si l'échelle temporelle entre chaque collaboration est relativement courte (chaque tour ne comprend plus que 5 minutes), pour évaluer la réelle simultanéité des événements, il faut que l'intervalle entre chaque intervention des différents acteurs soit au niveau de la milliseconde. Ainsi, pour les séances au Conservatoire, il ne s'agit pas du *live* au sens strict. Cependant, le résultat final a découlé d'un état d'esprit fortement fondé sur le *live patching*. Or, la notion du « commencer à zéro » est présente au-delà du développement du patch. Tout le projet a eu recours à cette méthode. Les modifications du projet original, à la fois techniques et musicales, ont émergé de notre pratique et de nos échanges. Cela prouve que les recherches au sein du projet ANR MUSICOLL ont dépassé le domaine du développement logiciel, elles ont surtout lancé une fertile perspective créatrice.

[8] http://www.youtube.com/mshparisnord/ (lien vérifié le 5 mai 2021).

5. REFERENCES

1. Bonardi, A., Roulleau, C., « Appropriations du langage FAUST, de la pédagogie à la recherche et la création en musique mixte, de l'université au conservatoire, sur le territoire de la Seine Saint-Denis », *Rencontres nationales sur les recherches en musique*, Ministère de la culture, 2020, Paris.

2. Bonardi, A. « An example of design of chains of delay lines in faust language for pedagogical and creation purposes in mixed music », *Proceedings of the International Faust Conference 2020 (IFC-20)*, 2020, Saint-Denis.

3. Collins, Nick, McLean A., Rohrhuber J., Ward A., « Live coding in laptop performance », *Organised sound* 8(3), 321-30, 2003.

4. Galleron, P., Maestri, E., Millot, J., Bonardi, A., Paris, E., « Enseigner le *patching* de manière collective avec le logiciel collaboratif Kiwi », *Actes des Journées d'informatique Musicale (JIM)*, Amiens, France, pp. 106-114, 2018.

5. Guillot, P., Paris, E., Bonardi, A., « Intégration du compilateur Faust dans l'application de patching collaboratif Kiwi », *Actes des Journées d'Informatique Musicale 2019*, Bayonne, 2019.

6. Magnusson, T., « Algorithms as scores: Coding live music », *Leonardo Music Journal* (21), 19- 23, 2011.

7. Messina, M., Svidzinski, J, Menezes Bezerra, D, Costa, D. F, « Live Patching and Remote Interaction: A Practice-Based, Intercontinental Approach to Kiwi », *14th International Symposium on Computer Music Multidisciplinary Research (CMMR),* Marseille, 2019.

8. Messina, M., Svidzinski, J, Costa, D. F, Silva, J. B. F, Martins, A. J, « Live/Acc/Patch: uma experiência coletiva internacional de programação musical ao vivo ». *South american journal of basic education, technical and technological*, v. 7, p. 1, 2020.

9. Paris, E., Millot, J., Guillot, P., Bonardi, A., Sèdes, A. « Kiwi : vers un environnement de création musicale temps réel collaboratif (premiers livrables du projet

MUSICOLL) », *Journées d'Informatique Musicale 2017*, Paris, France (2017).

10. Paris, E., *Une approche du patching audio collaboratif : enjeux et développement du collecticiel Kiwi*, Thèse de doctorat, Université Paris 8, 2018.

11. Ritsch, W., « ICE – towards distributed networked computermusic ensemble », In: Georgaki, A.,Kouroupetroglou, G. (eds.), *Proceedings ICMC|SMC|2014*, Athens, Greece, 2014.

12. Svidzinski, J., Bonardi, A**.,** « Reinjection matrices with faust language : creating complex structures in real-time mixed music »**,** *Proceedings of the International Faust Conference 2020* (IFC-20), 2020, Saint-Denis.

13. Svidzinski, J., Messina, M**.,** « Live patching with kiwi and Faust an intercontinental experience »**,** *Proceedings of the International Faust Conference 2020* (IFC-20), 2020, Saint-Denis.