FACULTAD
DE CIENCIAS
ECONÓMICAS

UNC Universidad Nacional de Córdoba

REPOSITORIO DIGITAL UNIVERSITARIO (RDU-UNC)

Proyectos tecnológicos innovadores en la producción de software: un estudio de caso

Hernán Alejandro Morero, Carina Borrastero, Jorge Motta

Capítulo del Libro XIX Reunión Anual Red Pymes Mercosur: Innovación, Desarrollo y conducta innovativa de las Pymes, 1º ed. publicado en 2014. ISBN 978-987-3608-07-0

Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional

Proyectos Tecnológicos Innovadores en la producción de Software: un estudio de caso

MORERO, Hernán Alejandro
hernanmorero@eco.uncor.edu
Centro de Investigaciones y Estudios sobre Cultura y Sociedad (CIECS), CONICET
Facultad de Ciencias Económicas – Universidad Nacional de Córdoba. Argentina.

BORRASTERO, Carina
carinaborrastero@conicet.gov.ar
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Argentina.

MOTTA, J.
jjmotta@eco.unc.edu.ar
Instituto de Economía y Finanzas – FCE –
– Universidad Nacional de Córdoba. Argentina.

1. Introducción y marco teórico de referencia

El objetivo general de este trabajo es analizar los procesos de innovación y desarrollo de capacidades en la actividad de Software y Servicios Informáticos (SSI) en la Argentina, a partir del estudio en profundidad de un caso particular. Los objetivos específicos de la investigación son: i) caracterizar el proceso de innovación en la actividad de producción de software a partir del análisis de proyectos tecnológicos innovadores que hayan impactado en el desempeño económico de la empresa; ii) examinar el papel que cumplen las competencias técnicas y organizacionales de la firma de software en sus procesos de innovación; iii) estudiar las particularidades que asume la organización del trabajo en los proyectos y su influencia en el desarrollo de capacidades y en los resultados de innovación; y iv) analizar las vinculaciones de esta empresa con el entorno y su incidencia en la propensión a innovar y las capacidades de la firma.

Para el estudio de caso se seleccionó una empresa de software de la ciudad de Córdoba (Argentina) con perfil innovador, denominada *Machinalis*. Al interior de la firma se estudiaron de modo cualitativo y comparativo las particularidades de dos proyectos tecnológicos innovadores.

Con este análisis, nos interesa contribuir al estudio de los procesos de innovación y desarrollo de capacidades en empresas de SSI reconociendo que la definición misma de

innovación en este tipo de firmas, así como las estrategias metodológicas de aproximación empírica, forman parte de un debate aún abierto.

Nuestro marco teórico de referencia está basado en aportes de la Teoría Evolucionista, del Institucionalismo, de la Economía de la Innovación y, particularmente, de la nueva literatura sobre Economía del Conocimiento (Ancori *et al.*, 2000, Cowan *et al.*, 2000, Ernst y Lundvall, 2004, Lundvall, 1996, Lundvall y Johnson, 1994, Lundvall, 1992, Nelson y Winter, 1982, Nonaka, 1995). Desde finales de los años '90 distintos autores se ocuparon de investigar en profundidad las características diferenciales de la innovación en los sectores de servicios, en comparación con las industrias manufactureras (Gallouj y Weinstein, 1997). En particular, los estudios en economía de la innovación tienden a resaltar ciertos rasgos propios de la producción de servicios que inciden en la naturaleza misma de los procesos de innovación (Drejer, 2004, Gallouj y Savona, 2009): la inmaterialidad del producto, una continua reconfiguración de la oferta, la co-producción y simultaneidad de la provisión y el consumo (Gallouj y Savona, 2009). En las últimas décadas se ha forjado un interés creciente por un tipo particular de servicio: los servicios intensivos en conocimiento o *Knowledge Intensive Bussines Services* (KIBS). Éstos se caracterizan por producir insumos inmateriales intensivos en conocimiento para los procesos de negocio en otras organizaciones que dependen fuertemente del conocimiento profesional (Miles, 2005, Muller y Doloreux, 2009). Entre ellos, el sector de SSI es uno de los más innovadores en países desarrollados y también en algunas economías emergentes (Niosi *et al.*, 2012). La producción de software es una actividad compleja que involucra un conjunto de fases altamente creativas y de servicios, en la que las actividades innovativas forman parte del propio proceso productivo. En su extremo más virtuoso, las conductas creativas de las firmas serán las que habiliten la emergencia de innovaciones, en oposición a las conductas de tipo adaptativo (Antonelli, 2008, Schumpeter, 1947). Ello tanto en términos estrictamente técnicos, como organizacionales y comerciales.

En este marco cobra relevancia un estudio que profundice en la naturaleza misma del proceso innovativo en empresas de SSI y en la dinámica de su integración al proceso de producción. Por otro lado, una mejor comprensión de la naturaleza de la innovación puede generar aportes significativos al diseño de instrumentos de medición de la innovación que se encuentran en diversos estadios de desarrollo y prueba. Por ello, este trabajo propone una investigación cualitativa a través del estudio comparativo en profundidad de dos proyectos tecnológicos exitosos e innovadores, que permite indagar *cómo* se da el proceso innovativo

en este tipo de proyectos en una empresa dinámica que opera en un segmento *top knowledge* del sector.

El trabajo está movilizado por una serie de interrogantes en torno a este planteo: ¿Qué aspectos internos y externos de la firma inciden en la realización de un proyecto innovador en una empresa de software, así como en su éxito o fracaso? ¿Qué papel cumplen las capacidades organizacionales y tecnológicas de la empresa en el desarrollo de proyectos innovadores? ¿Qué aprendizajes surgen del desarrollo de un proyecto innovador y cómo inciden en la dinámica de la firma? Y en términos más generales: ¿Qué caracteriza a una empresa innovadora del sector de SSI? ¿Cuál es la dinámica de desarrollo de capacidades en este tipo de firmas? ¿Cuál es el rol de las vinculaciones con otros actores en el proceso innovador de una empresa de SSI? ¿Cuáles son los determinantes de la innovación en PYMES de software?

El artículo se organiza del siguiente modo. En la sección 2 se presenta el marco metodológico de la investigación. En la sección 3 se presenta el caso a partir de la descripción exhaustiva de las características de los proyectos tecnológicos seleccionados, sistematizadas en torno a las variables relevantes surgidas del estudio: orígenes y fases de desarrollo del proyecto; organización del trabajo al interior del proyecto; aprendizajes, capacidades y vinculaciones; medición de la *performance* y resultados de innovación obtenidos. En la sección 4 se realiza un análisis comparativo de los proyectos tecnológicos en torno a las variables seleccionadas. Por último, en la sección 5 se presentan algunas reflexiones finales que pueden orientar futuras líneas de indagación.

2. Metodología

La estrategia metodológica es cualitativa. Se propone un estudio exploratorio y descriptivo, especialmente indicado para conseguir una familiarización con un fenómeno aun escasamente comprendido, para generar nuevas ideas que permitan formular preguntas y nuevas hipótesis y conocer la perspectiva de los sujetos protagonistas bajo estudio (Denzin y Lincoln, 2005, Eisenhardt, 1989, Neergaard y Ulhøi, 2007, Yin, 2009). En ese sentido, se plantea el estudio de un caso en profundidad para indagar acerca del “cómo” se da el proceso innovativo en una empresa de producción de SSI.

El estudio expone el análisis de un caso único, tomando como unidad de análisis los proyectos tecnológicos de la empresa. Según Yin (2009) el diseño para el estudio de un caso único es pertinente para un caso crítico (*critical case*), un caso extremo (*extreme case*), casos típicos (*typical cases*), un caso revelatorio (*revelatory case*) o un caso longitudinal. Se justifica la elección de un *typical case* cuando el objetivo es *"capturar circunstancias y condiciones de una situación común y cotidiana"* y el caso *"puede representar un 'proyecto' típico dentro de muchos proyectos diferentes (…)"* (Yin, 2009). Por su parte, un caso revelatorio es pertinente *"cuando el investigador tiene la oportunidad de observar y analizar un fenómeno previamente inaccesible para la investigación en ciencias sociales"* (Yin, 2009). Según Neergaard (2007) los casos revelatorios pueden ser vistos como una sub-especie de los *typical cases*.

De este modo, se buscó una empresa dinámica del sector de software que opere en un segmento '*top knowledge*' del mercado, que ofrezca *insights* tanto como *typical case* respecto a las características del proceso de innovación en firmas del sector y que sea de elevada complejidad tecnológica, y a la vez dos proyectos tecnológicos innovadores con rasgos distintivos, que ofrezcan potencialidades asimismo como *revelatory case*. Ello en tanto esta estrategia brinda la oportunidad de observar y analizar la naturaleza de la innovación en este tipo de sectores a través del estudio de proyectos como unidad de análisis, en lugar de la empresa que es el nivel de análisis más usual en la literatura de referencia.

A partir de la realización de entrevistas con informantes clave de la firma y la revisión de documentación, el análisis se centró en el estudio en profundidad de dos proyectos tecnológicos relevantes para la dinámica innovadora y económica de la empresa. Machinalis se caracteriza por realizar dos grandes tipos de proyectos tecnológicos:

- Proyectos comerciales (*client led projects*): se trata de desarrollos a medida típicamente comerciales, orientados a la satisfacción de las necesidades de un cliente.
- Proyectos internos (*internally led projects*): se trata de desarrollos no orientados a clientes ni de objetivos estrictamente comerciales, de tipo *open source*.

En virtud de esta particularidad de la empresa y con el objeto de obtener una mayor riqueza interpretativa, se seleccionaron dos proyectos tecnológicos representativos de cada tipo:

- El proyecto Proflow, que consiste en el desarrollo de un software de gestión integral para la industria de tratamiento de aeropartes. El rasgo distintivo de Proflow es tratarse de un proyecto de tipo *client led*.
- El proyecto *Quepy*, que consiste en el desarrollo de un software conversor de preguntas en lenguaje natural a *queries* de bases de datos. El rasgo distintivo de Quepy respecto a Proflow es el tratarse de un proyecto interno de la empresa.

Las unidades de análisis fueron estudiadas diacrónicamente en función de cuatro ejes de análisis: su origen y las fases de su desarrollo, las particularidades que asume la organización del trabajo, los aprendizajes surgidos del proyecto y el desarrollo de capacidades y vinculaciones, y las medidas de *performance* y resultados de innovación. Se realizaron 8 entrevistas semi-estructuradas alrededor de estos ejes de una duración de entre 40 y 90 minutos cada una a distintos integrantes de la empresa: Director Ejecutivo (o CEO, por *Chief Executive Officer*), Responsable Coordinador de Proyectos, *Project Managers*, *Technical Leaders* y *desarrolladores*.

A partir del estudio diacrónico, se analizaron en términos comparativos las características de los proyectos relativas a las variables de análisis mencionadas desde la perspectiva de los procesos de innovación en la firma.

A lo largo de la reconstrucción del caso se mencionará a los integrantes de la empresa por su nombre de pila y la inicial de su apellido sumado a su puesto en el proyecto del que forman parte. Al cliente nos referimos mediante el seudónimo X. para mencionar a la empresa y J. para mencionar a su dueño/CEO/jefe de operaciones según corresponda.

3. Los proyectos innovadores de Machinalis

Machinalis es una empresa de la ciudad de Córdoba, Argentina, especializada en el desarrollo de software y servicios en las áreas de Inteligencia Artificial (en particular, *Natural Language Processing*, *Data Mining* y *Machine Learning*), Data Processing, Big Data o Data Science (incluyendo *Extraction, Data Analysis, Data Modelling* y *Data Visualization*) y Servicios de Desarrollo Complejo (*Complex Web Development*, *Desktop Development* y *Process Automation*).

La firma se dedica principalmente al desarrollo de software a medida, tiene una antigüedad de 4 años, exporta el 100% de su producción y cuenta en la actualidad con una plantilla de

alrededor de 35 empleados. Entre ellos, la mayoría tiene un alto nivel de calificación formal, principalmente educación universitaria completa (el 65% de la planta), y posgrado completo (el 15% de los empleados).

A continuación reportamos los resultados obtenidos del relevamiento de los proyectos que conforman la unidad de análisis del presente trabajo.

3.1. Un proyecto para clientes: Proflow

Proflow es un sistema de gestión integral desarrollado a medida por Machinalis para una empresa estadounidense de tratamiento de aeropartes. Le permite al cliente manejar toda su información desde los pedidos de presupuesto, producción, facturación, entrega, etc. Todos los procesos de negocios están automatizados. Permite a todos los sectores de la empresa monitorear en qué parte del proceso está cada orden de trabajo, lo que principalmente se traduce en ahorro de tiempo en cada operación[9].

- *Origen y fases de desarrollo del proyecto*

El proyecto surgió a partir de una persona conocida de un amigo del actual CEO de Machinalis, que tiene un hermano en los Estados Unidos que era dueño y también CEO de X. en aquel momento (J.). J. estaba buscando desarrollar un sistema de información a medida que permitiera agilizar los procesos de la planta, dada la inexistencia de un software específico para el segmento productivo del cromado y tratamientos químicos para aeropartes[10]. La mencionada persona en común conectó a ambos CEOs como posible vía para satisfacer esta necesidad. J. decidió contactar a Machinalis porque reconocía que escalar el software genérico que venía usando hasta el momento resultaría igual de costoso que desarrollar uno nuevo que le daría a la empresa muchas más ventajas. Fue la empresa de J. la que realizó el estudio previo y llegó a la conclusión de que no existía un software adaptado. Las opciones para el desarrollo a medida eran Machinalis, otra empresa de Argentina y una de Colombia. J. planteó que necesitaba una empresa que les hiciera un sistema de gestión totalmente adaptado y que pudiera continuar desarrollando según las nuevas necesidades que fueran surgiendo, y proveyéndoles el soporte. Según los entrevistados, la elección de Machinalis surgió de un proceso típico comercial, donde el

[9] Por ejemplo: la atención telefónica al cliente antes insumía una media hora y con el software las respuestas pueden darse instantáneamente; la planificación de las órdenes de producción tardaba unos cinco días y con el software se realiza en dos horas aproximadamente.

[10] Hasta el surgimiento de Proflow no existía un software específico para esta industria. Las empresas utilizaban software de gestión estándar, tipo Tango, sobrellevando sus limitaciones.

cliente evaluó que el proyecto presentado sería el menos costoso. A partir de este primer contacto y luego de 15 días, quien sería el primer *Project Manager* de Proflow viajó a EEUU para visitar la planta e identificar qué era exactamente lo que el potencial cliente necesitaba. A partir de esa entrevista inicial comenzaron a relevar los primeros requerimientos de software, y a partir de un documento elaborado por el *Project Manager* para presentar en Machinalis surgió el proyecto propiamente dicho, se conformó un grupo de trabajo y comenzaron las tareas de desarrollo. Fueron tres las razones principales que impulsaron a Machinalis a definir desarrollar el proyecto: la última etapa programada del proyecto involucraba originalmente el desarrollo de herramientas basadas en los campos de *expertise* de la empresa y en ese sentido resultaba desafiante según destacan los entrevistados; se trataba de un proyecto que generaría ingresos importantes; y permitiría la inserción en el mercado estadounidense, objetivo de la empresa.

La primera fase del proyecto Proflow constituyó el desarrollo del sistema de gestión integral en sí, y duró dos años. Según los entrevistados, el periodo inicial de esta fase fue fundamental para lograr el objetivo *sine qua non* de conocer la industria de destino, hasta ese momento completamente ajena a la experiencia de la empresa:

> *No hay forma de construir un software hoy en día sin tener un contacto directo con el cliente. (…) Porque incluso los clientes tampoco saben qué es lo que necesitan, tienen una visión, pero parte de nuestro trabajo es bajar esa visión a tierra, desmenuzarla y proponerle al cliente, decirle 'mirá: ésto es lo que querías y no lo que me proponías al principio'. Entonces ese proceso de conocimiento se da cara a cara con reuniones*[11].

Una vez listo el núcleo del software, el proyecto entró en una segunda fase que consistía en el desarrollo de herramientas de monitoreo de la producción, que duró aproximadamente un año (los primeros seis meses dedicados al desarrollo intensivo de las herramientas y los siguientes seis meses a los ajustes).

Las dos primeras fases involucraron una inversión de recursos humanos de entre 4 y 10 personas dependiendo del periodo específico: el periodo que más recursos demandó fue el segundo semestre de 2012 durante el cual trabajaron 10 personas, y en el periodo en que menos trabajadores se involucraron el equipo estaba integrado por 4 personas (2° semestre

[11] Entrevista a Patricio D.B., *Project Manager* de Proflow, 22/05/2014.

de 2013). En promedio, el tamaño del equipo osciló entre 5-6 y 9-10 personas. En la trayectoria de Proflow, casi todos los empleados de la empresa integraron el proyecto en algún momento. A la fecha del trabajo de campo (junio de 2014) 6 personas lo llevaban adelante. Se trata del proyecto de mayor envergadura que la empresa ha realizado desde su nacimiento.

Al iniciar la relación con el cliente se planificó una tercera fase que hasta el momento de elaboración de este artículo no se ha llevado a cabo. Dicha fase implicaría incorporar *Machine Learning*, Inteligencia Artificial, Redes neurales y todos los campos *core* de conocimiento de la empresa al desarrollo de una herramienta que permita medir la "capacidad de carga" del taller.

> *Hoy en día no hay una medida en la industria que a ellos les permita saber cuál es su cuello de botella, dónde están mis mayores problemas, cuántas órdenes debería ser capaz de procesar un operador por día. Esas respuestas, por las características de la industria, hoy en día no están respondidas* [sic]*, no hay conocimiento de eso. Estamos tratando de lograrlo, es complicado, pero estamos tratando. Y esta es la segunda ventaja estratégica o innovadora del sistema, otro de los objetivos que tenía J. cuando quiso empezar este sistema*[12]*.*

Desde la visión de los entrevistados se trataría de la fase más innovadora del proyecto en términos del conocimiento existente en estos campos de la computación y su aplicación a la industria de destino. Insumiría alrededor de siete meses y unos 4 o 5 desarrolladores[13]. La suspensión de esta fase tiene que ver con lo que sigue. La empresa cliente fue adquirida por un fondo de inversión, que compró a la vez otras plantas similares en distintas localizaciones de EEUU. Así, cambió la organización administrativa y el antiguo dueño (J.) pasó a ser accionista y jefe de producción en la planta original. Los nuevos dueños realizaron una auditoría del software, comparativa con el software genérico que utiliza la competencia, y definieron que Proflow era el más adecuado, por lo que le dieron continuidad. Por lo tanto, el sistema originalmente diseñado para una sola planta debe ser adaptado al uso simultáneo en varias plantas, lo que implica migrar el sistema a otras arquitecturas de software y permitirle centralizar toda la información a partir de una tecnología de sistemas distribuidos.

[12] Ibídem.
[13] Su desarrollo efectivo está pensado para el segundo semestre del corriente año.

De allí que la tercera fase originalmente ideada quedara postergada por esta última tercera fase real, dado el cambio en las necesidades de negocio del cliente.

Organización del proceso de trabajo

Para el desarrollo y mantenimiento del sistema se lleva a cabo un proceso de ingeniería de software "iterativo e incremental". El software se implementa por partes o periodos denominados iteraciones o *sprints*, que el cliente prueba y evalúa. Una vez completada cada iteración, se pasa a la siguiente. No se realiza de una vez todo lo que el cliente necesita. Las iteraciones planificadas originalmente pueden variar en tiempo y forma de acuerdo a la marcha del proceso, hay flexibilidad. Las etapas básicas del proceso son: análisis de requerimientos (establecer con precisión y en términos técnicos qué quiere el cliente del software), desarrollo, prueba e implementación. Esas actividades se iteran todo el tiempo (una iteración dura aproximadamente dos semanas). Ante el inminente inicio de cada iteración se realiza una estimación colectiva del tiempo que ésta insumirá, a través de la técnica del *poker planning*[14]. Y una vez finalizada cada iteración se realiza una "reunión de prospectiva" en la que se analiza colectivamente qué se hizo, resultados y acciones de mejora constante. En suma, el proceso es incremental porque se va implementando de a poco, y de a poco va creciendo la solución. Para llegar a las soluciones utilizan la dinámica del *brainstorming* entre todos los integrantes del equipo.

Para organizar los procesos de trabajo emplean Metodologías Ágiles tipo *Scrum*[15]. Estas Metodologías Ágiles involucran el tipo de proceso descripto arriba, y prevén diversos roles y funciones al interior del equipo de trabajo, que en Proflow se plasmaron del siguiente modo:

- Dueño del producto (*Product Owner*): es quien define las especificaciones que tiene que tener el producto y una lista de prioridades, que en este caso es el cliente.
- *Project Manager*: gestiona el proyecto. Es el encargado de comunicarse con el *Product Owner*, estructurar los requerimientos del software en forma clara, comunicarlos al equipo de desarrollo, tomar decisiones de organización, coordinar reuniones, oficiar de nexo entre el cliente y el equipo de desarrollo y entre éste y las demás áreas de la empresa que lo requieran. Cumple una función de "protección" del equipo de las presiones externas para que los desarrolladores puedan trabajar con

[14] Mediante naipes especialmente numerados, cada integrante del equipo propone una estimación. Cuando éstas tienden a coincidir se establece un promedio colectivo, y si existen diferencias significativas se discute y a partir de las visiones planteadas se realiza nuevamente el juego.
[15] Ver (Sutherland y Schwaber, 2012, Takeuchi y Nonaka, 1986)

tranquilidad en sus tareas específicas (ver más abajo). El rol de *Project Manager* no requiere necesariamente el mismo nivel de competencias técnicas que el resto de los integrantes del equipo, sino más bien de competencias "manageriales" que aporten dinamismo al proceso de desarrollo, como capacidades de comunicación, manejo de grupos, administración de tiempos de trabajo.

- *Technical Leader*: su rol principal es detectar riesgos tecnológicos, y eventualmente diseñar la arquitectura del sistema. Toma decisiones relevantes a nivel tecnológico pero a partir de soluciones propuestas por el equipo de desarrollo, que en general se consensúan.
- Equipo de desarrollo: tiene una autonomía relativa para asignar las tareas hacia adentro. Las funciones generales del desarrollador (*developer*) son el desarrollo del sistema y codificación de pruebas automáticas, pero puede asumir roles diversos como Desarrollador *Front-End*, Desarrollador *Back-End*, Administrador de Base de Datos, *Sysadmin*, *Tester*.

La empresa ha realizado adaptaciones de la metodología *Scrum* en función de ciertos rasgos de la cultura organizacional de Machinalis que *"chocaban con ese framework"*[16], que impactan principalmente sobre los roles y funciones clásicos de los integrantes del equipo de trabajo:

- En las Metodologías Ágiles el *Product Owner* tiene contacto directo con el equipo de desarrollo. En Proflow sólo el *Project Manager* habla con el equipo, dado que el *Product Owner* es el cliente.

> *El Project Manager trata de proteger un poco al equipo para que el equipo esté tranquilo (…). Regula el flujo de información y conecta con el equipo y controla (…). Por ahí porque la cultura de Machinalis es una cultura mucho más técnica, que no tienen por ahí las habilidades blandas muy desarrolladas, por decirlo así, entonces por ahí les cuesta hablar con el cliente*[17].

También aclaran que ese criterio es útil particularmente cuando el *Product Owner* no sabe (o no mucho) de software[18] y es necesario un nexo que garantice la idoneidad de las decisiones técnicas.

[16] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014.
[17] Ibídem.
[18] Las Metodologías Ágiles *Scrum* presuponen lo contrario.

- En las Metodologías Ágiles tradicionales no hay *Project Manager* entendido como responsable máximo del proyecto, el equipo es auto-organizado, y existe una figura de facilitador (*Scrum Master*) que ayuda a que el equipo sea más eficiente pero no toma decisiones hacia los demás.
- El Project Manager de Proflow explicó que en un primer momento del desarrollo de las metodologías ágiles se pensaba al *Project Manager* y al *Technical Leader* como

  'Gurúes que saben mucho'. Pero en esta empresa hay mucha gente que sabe mucho. Después hay proyectos donde todos saben más o menos lo mismo, entonces hay proyectos como el nuestro donde el rol del Technical Leader se decide por cuánto más conocés del proyecto[19]*.*

- Desarrolladores: en Proflow trabajan desde la premisa de un *"equipo multifuncional donde todos hacen todo"*[20], sin diferenciaciones rígidas entre las diversas tareas que pueden corresponder a un desarrollador (mencionadas más arriba) con el objeto de evitar la interrupción de tareas.

  Perdemos un poco de especialización, no siempre la interfaz gráfica es bonita y muchas veces el código back-end no es prolijo, pero la gestión y el desarrollo es mucho más fluido. Eliminamos interdependencias[21]*.*

Los entrevistados aclararon que *Scrum* siempre requiere adaptación, que toda empresa adapta esas buenas prácticas a su realidad.

La gran mayoría de los integrantes del equipo de Proflow han tenido/tienen título universitario o estudios avanzados[22]. En las tareas de desarrollo predominan los profesionales de Ingeniería o Licenciatura en Computación, y en las tareas más ligadas al *management* los Ingenieros en Sistemas. Los no graduados tienen el mismo tipo de participación en el proyecto, la titulación no es determinante en la asignación de tareas ni responsabilidades, en

[19] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014
[20] Ibídem.
[21] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014
[22] Ingenieros en Computación o en Sistemas, Licenciados en Computación, estudiantes avanzados de esas carreras.

general ello tiene más que ver con la edad y la experiencia que con el grado de calificación formal[23].

- *Aprendizajes, desarrollo de capacidades y vinculaciones*

Los entrevistados destacaron que el efectivo desarrollo de Proflow, dada su envergadura y la inexperiencia previa de la empresa en la industria de destino del sistema, fue posible, por un lado, gracias a un cúmulo de capacidades preexistentes, relacionadas principalmente con las competencias de los recursos humanos en las áreas de desempeño de la empresa.

> *Somos una empresa que se especializa en Python y Django, entonces acá hay gente reconocida mundialmente por su trabajo en Python y por su contribución a la comunidad de Python*[24]*.*
>
> *Para nosotros tener científicos dando vuelta por acá es habitual. Si no tengo el científico acá no puedo seguir*[25]*.*

El *Project Manager* señala estos elementos como diferenciales de Machinalis respecto a otras empresas de software de Córdoba.

Una vez en marcha Proflow, en visión retrospectiva los entrevistados señalan una variedad de aprendizajes derivados del proyecto, asociados principalmente con aspectos organizacionales y en menor medida comerciales.

En relación con el primer aspecto se destacan, por un lado, las adaptaciones de las Metodologías Ágiles realizadas durante el desarrollo del proyecto que ya mencionamos en el apartado anterior, fruto de las dificultades de organización con las que el equipo se fue encontrando.

> *Cuando se comenzó con este sistema, como que no estaba muy claro estas metodologías ágiles, entonces había una persona que era el líder técnico, y además era Project Manager, además hablaba con el cliente, además era desarrollador, hacía todo, y duró cuatro meses. Entonces creo que un aprendizaje mayor fue cuando se ingresó un Project Manager al proyecto. (…) Ése fue el principal punto de inflexión, cuando se dieron cuenta que había un rol que no era técnico que servía. (…) Incluso*

[23] Ninguno de los dos entrevistados del proyecto, por ejemplo, estaba graduado al momento de asumir sus puestos actuales en el proyecto (*Project manager* y *Technical Leader*).
[24] Entrevista a Patricio D.B., *Project Manager* de Proflow, 22/05/2014.
[25] Ibídem.

> *después de ver en Proflow que el rol de Project Manager funcionaba y era crítico se creó una política organizacional entre comillas de decir 'todo proyecto tiene que tener un Project Manager, nunca más ponemos a un desarrollador al frente del cliente'*[26].

> *Ahora el otro aprendizaje grande que está teniendo Proflow es que el rol del Project Manager está muy saturado. Hoy en día tiene muchísimas tareas para hacer, entonces están empezando a ver dentro del rol de Project Manager cuáles son los roles y empezar a dividirlo en distintas personas*[27].

Además, los entrevistados destacan que a partir del desarrollo de Proflow, por su envergadura, la empresa aprendió a reconocer las potencialidades y límites del tamaño del equipo de desarrollo.

> *Creo que de eso está bueno aprender, hasta dónde puede llegar el equipo, qué tan grande puede ser el equipo, porque tampoco podés tener tantos desarrolladores, si tenés muchos desarrolladores empezás a tener mucha incertidumbre, y la iteración va a abarcar más tareas, y muchas tareas son interdependientes (…), qué tan larga puede ser una iteración, qué tan malo puede ser que se extienda. (…) Por ahí el cliente te pide capacidad de desarrollo y vos decís 'uno más, uno más', y llega un momento en que con ese uno más termina todo así* [dado vuelta][28].

> *Cuando llega ese límite, otro equipo. Hay que hacer otro equipo con las mismas características. Los dos equipos trabajan sobre la misma pila de tareas pero se organizan independientemente. (…) A las dos semanas se vuelven a juntar, y se separan, y así*[29].

Otro de los aprendizajes organizacionales relevantes fue que el equipo determine la duración de las iteraciones según sus capacidades y ya no el cliente, a partir de haber aceptado una imposición del cliente al respecto que resultó en un fracaso para el avance del proyecto[30]. Esto se estableció como una "regla" en la empresa a partir de allí. Al mismo tiempo este aprendizaje de tipo organizacional condujo a establecer ciertas modificaciones en la gestión de los proyectos desde el área comercial: la propuesta fue del equipo de Proflow al área

[26] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014
[27] Ibídem.
[28] Entrevista a Francisco, *Technical Leader* de Proflow, 24/06/2014.
[29] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014.
[30] Relatan que el cliente pidió *sprints* de 45 días, lo que generó que las desviaciones en las tareas respecto a las estimaciones fueran altísimas. *Sprints* de 15 días permiten a la empresa tener una mejor revisión de las estimaciones.

comercial de la empresa, y a partir de ese momento cuando esa área negocia con el cliente acuerda específicamente que este último no va a decidir sobre ese punto.

Los entrevistados destacan que el proceso permanente de ensayo y error es una de las fuentes principales del tipo de aprendizajes mencionados, y que ello es propio, a su vez, de las Metodologías Ágiles que implementan para organizar el trabajo al interior de los proyectos.

> *Básicamente somos una empresa que es nueva, entonces estamos todo el tiempo chocándonos contra la pared, siempre cometemos errores y tratamos todo el tiempo de tratar de ver cuáles son los errores que cometimos y tratar de ver de que la empresa en futuros proyectos no vuelva a cometer los mismos errores*[31]*.*

En relación con esto último, los entrevistados enfatizan que este tipo de metodologías de desarrollo de software también permiten aprender al nivel de los procesos empresariales en general. En este sentido relatan que actualmente están encarando un proyecto interno que reúne a todas las áreas de la empresa, destinado a reducir el tiempo de desarrollo en base a los aprendizajes surgidos de todos los proyectos realizados.

Una fuente muy clara de aprendizajes y desarrollo de nuevas capacidades en la empresa a partir de Proflow es la vinculación con el cliente. Éste parece haber tenido en este caso un papel clave en las condiciones para innovar en el desarrollo del software por tratarse de una empresa también innovadora al interior de su rama industrial.

> *Entonces no había en el mercado un software ajustado a esta industria. La competencia utiliza este otro sistema, que no tiene las prestaciones de Proflow, pero como el estándar sabe que medianamente funciona, lo usa. (…) Pero J., que es un poco más visionario, quería algo que le dé una ventaja estratégica un poco más importante, entonces quería desarrollar uno que le quedara como guante a la industria*[32]*.*

También es importante señalar que, desde la visión de los entrevistados, esta vinculación fue clave por haberse producido una dinámica de aprendizajes e innovaciones en la que ambas empresas adquirían mayor protagonismo en diversas etapas del desarrollo del proyecto según las necesidades y conocimientos que cada una manifestaba y podía aportar. Al mismo

[31] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014.
[32] Entrevista a Patricio D.B., *Project Manager* de Proflow, 22/05/2014.

tiempo, destacan que esta dinámica le permite al cliente innovar en sus propios procesos y correr el límite de sus propios conocimientos.

> *Básicamente, todas estas nuevas herramientas cambian completamente la forma de trabajo de ellos. Entonces es un trabajo que tenés que hacer en conjunto con el cliente porque vos podés innovar para la parte de software, pero vos necesitás sí o sí alguien del lado del otro que identifique como estás innovando con el software y te diga 'entonces yo puedo innovar de esta forma en mis procesos'. Así que sí, es una innovación por ambos lados*[33].

A su vez, el tipo de relación establecida con el cliente y el devenir del proyecto[34] han implicado la necesidad de establecer vinculaciones con otros agentes que constituyen fuentes de conocimiento externas de relevancia. Respecto al desarrollo de herramientas para medir la capacidad de carga del taller, si bien fue suspendido por el cliente, Machinalis continúa trabajando en esta dirección independientemente del cliente pero bajo su conocimiento, y para ello se ha vinculado con un espacio académico especializado del extranjero.

> *Machinalis se puso en contacto con el centro de modelado matemático del laboratorio de análisis combinatorio de la Universidad de Chile, porque ellos son personas que se especializan en modelar procesos productivos. Y ver si entre ellos* [el cliente], *nosotros y el laboratorio de Chile somos capaces de armar algún modelo matemático que nos permita optimizar eso. (…) Y ese fue el momento más complicado, desde que yo estuve, porque ahí es donde más tuvimos que innovar, ahí es donde más tuvimos que salir a decir 'tenemos estos problemas, tenemos esta gente, tenemos esta herramienta, esto tiene que seguir andando, muchachos ¿cómo hacemos para seguir andando? Bueno… llamá a la universidad de Chile'* [35].

El CEO de Machinalis decidió iniciar la relación con el laboratorio chileno sin tener muchas certezas sobre los resultados: *"Al menos servirá para probar y decir 'por ahí no es'"*[36].

[33] Ibídem.
[34] En concreto, la suspensión de la tercera fase originalmente planteada y su reemplazo por otra con características diferentes.
[35] Entrevista a Patricio D.B., *Project Manager* de Proflow, 22/05/2014.
[36] Ibídem.

Además del laboratorio de Chile, en la actualidad Machinalis está en conversaciones con una empresa de Dallas que se especializa en sistemas distribuidos buscando que ésta les provea el soporte para poder diseñar la arquitectura del nuevo software para varias plantas[37].

Son muy frecuentes e intensas también las vinculaciones de Machinalis con las comunidades tecnológicas de referencia del *framework* de desarrollo (Django[38]) y el lenguaje de programación en que se especializan (Python[39]), lo que se ha manifestado de igual modo durante el desarrollo de Proflow.

Una última fuente de aprendizajes que destacan los entrevistados es la aplicación de normas de calidad ISO 9001 en los procesos de gestión.

> *Esta certificación se realizó cuando el proyecto llevaba poco más de un año. Lo valioso de la certificación es que nos permite detectar oportunidades de mejoras y mejora continua a los procesos de gestión y procesos de desarrollo del proyecto. (…) Hoy en día, el proceso de gestión de proyectos de la empresa está muy embebido de experiencias y prácticas realizadas en el proyecto*[40]*.*

- *Resultados de innovación y medidas de performance.*

La medición de la performance del proyecto es interna, los integrantes del equipo de trabajo desconocen cómo impacta el proyecto en la productividad y el desempeño económico de la empresa. Cada vez que comienza una iteración se elabora una lista de tareas a realizar y cada tarea tiene una medida de esfuerzo, que en el caso de Proflow se traduce en tiempo (horas y días). Para monitorear su cumplimiento post-iteración utilizan un gráfico llamado *burndown*, propio de las Metodologías Ágiles, que permite medir la productividad dentro de la iteración: la distancia existente entre lo efectivamente realizado y la velocidad ideal de trabajo del equipo. A nivel del proyecto en general aplican la misma medida pero en términos

[37] Al momento de elaboración de este artículo dicha vinculación estaba en sus comienzos por lo que no contamos con mayor información ni evaluaciones al respecto.

[38] Un *framework* es una estructura conceptual y tecnológica de soporte para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas. Django es un *framework* de desarrollo web de código abierto, escrito en el lenguaje de programación Python, que respeta el paradigma conocido como *Model Template View*. La meta fundamental de Django es facilitar la creación de sitios web complejos, poniendo énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio "no te repitas" (DRY, del inglés Don't Repeat Yourself). Más información en https://www.djangoproject.com/.

[39] Python es un lenguaje de programación con licencia de código abierto, compatible con la licencia pública general de la GNU. Más información en http://www.python.org/.

[40] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014.

"macro": trabajan por objetivos más que por tiempos y realizan un seguimiento continuo muy similar de su cumplimiento.

La cuestión de la medición de la *performance* es relevante para la empresa, en relación a su vez con el tipo de organización del trabajo que practican:

> *El gran problema que tenemos en el software es la estimación. El cliente viene y te dice 'cuánto me lleva hacer esto', y no hay forma, en software es casi imposible de determinar. Entonces vos nunca te comprometés a más de dos semanas*[41]*.*

> *Hay como una sobrecarga en la estimación. Te lleva tiempo. Por ahí de diez días, capaz que medio día nos toma estimar las tareas*[42]*.*

Los integrantes del proyecto sí acceden a información de *performance* del cliente, que les provee el propio software, y de este modo la empresa puede ir siguiendo y evaluando qué cambios ha producido el sistema en relación con sus funcionalidades. Según los entrevistados, dichos cambios se producen a nivel de la productividad y también en términos de la cultura organizacional de la empresa cliente.

Los resultados de innovación efectivamente obtenidos durante el desarrollo de Proflow reúnen principalmente: a) las innovaciones organizacionales ya referidas, b) el sistema en sí, y c) innovaciones técnicas que forman parte del producto pero no son necesariamente visibles aunque sí se socializan en las comunidades tecnológicas.

> *Creo que somos de alguna forma pioneros en atacar el problema de este tipo de industria específica, que es orientada al shop, una línea de producción de cosas que entran por allá y salen por acá*[43]*.*

En particular, señalan que lograron realizar las dos primeras fases del proyecto gracias al desarrollo de un algoritmo de árboles de decisión. Según explican, ello constituye una innovación técnica en tanto:

> *Es innovarlo… estás innovando porque lo estás utilizando para esta industria, en cierto punto. O sea, los algoritmos de árboles de decisión existen, pero haberse dado cuenta de que un algoritmo de árbol de decisión le iba a permitir a este Planificador,*

[41] Ibídem.
[42] Entrevista a Francisco, *Technical Leader* de Proflow, 24/06/2014.
[43] Ibídem.

> *en base a todas las características de esta industria, poder hacer una decisión rápida de qué proceso hacer con la pieza, eso no, no existía antes, entonces Machinalis se dio cuenta de esos nexos, por decirlo así*[44].

En segundo lugar comentan que un proceso similar ocurrió en torno a las tecnologías que utilizan para desarrollar el software: por un lado el *framework* web Django que combinan con otras tecnologías que suplen las herramientas que van faltando.

> *Nosotros expandimos, llegamos a los límites de esta herramienta. (…) Tuvimos que mover la línea de esa herramienta. (…) Entonces hoy en día no hay un producto hecho así con Django. (…) Y hoy en día estamos utilizando, no sé, ocho tecnologías distintas para hacer esto. (…) Y todas estas, decimos, de alguna forma meterlas adentro de Django, y eso es bastante innovador en cuanto a Django*[45].

Por otra parte introdujeron, a partir de la combinación de distintas tecnologías, una innovación útil para lograr extraer un gran volumen de información del sistema por parte de muchos usuarios simultáneamente, problema que no estaba resuelto previamente y resultaba imprescindible para determinadas operaciones en la planta.

> *Cosas que antes hubieran sido imposibles, al menos según lo que estaba en la red y lo que fuimos encontrando, le dimos esa vueltita de rosca*. (…) *Juntamos cosas que no estaban juntas antes (…). Eso nos habrá llevado más o menos un mes, pero la verdad que tuvimos resultados muy muy buenos* [sic]. *(…) Nosotros trabajamos con modelos. Los modelos son todos conocidos… quizá en los que fuimos originales nosotros es en usarlos para esta industria, para este problema en particular*[46].

El *Project Manager* explicó que a ese problema la comunidad mundial de Django no lo había solucionado. Por último, destacan que introdujeron también innovaciones técnicas en el sistema operativo Linux para resolver un problema operativo del taller (necesidad de uso de "terminales bobas").

> *Dentro del universo de posibilidades que había acá, seleccionamos un par que nos servían, las mezclamos, las combinamos, y las añadimos a Django. De hecho Django, que era un framework de desarrollo web que no estaba preparado para*

[44] Entrevista a Patricio D.B., *Project Manager* de Proflow, 22/05/2014.
[45] Ibídem.
[46] Entrevista a Francisco, *Technical Leader* de Proflow, 24/06/2014.

> *recibir terminales bobas, de repente nosotros juntamos unas cuantas herramientas que daban vueltas por la red y toqueteamos el sistema operativo, logramos que el sistema operativo dijera 'dame lo que quiero'*[47].

El *Project Manager* explicó que esa innovación puede ser aplicable a cualquier sistema (ej. el sistema de un hospital).

Por último, respecto a las posibilidades de liberar los resultados de innovación obtenidos durante el desarrollo de Proflow, hay evidencia de cierta tensión entre la propiedad privada del software cuando el proyecto es para un cliente y el impulso a la innovación que motiva a la empresa. Cuando se realiza un desarrollo a medida para un cliente, la propiedad intelectual, por contrato, pertenece a este último. De todas maneras, cuando existen innovaciones relevantes que no hacen a la esencia del software, Machinalis tiende a liberarlas entendiendo que de ese modo se retribuye a su vez a la comunidad tecnológica por herramientas libres que también utilizan a beneficio del cliente sin que éste las pague o siquiera las conozca.

3.2. Un Proyecto Interno: Quepy

Quepy es un *framework* de *Python* para transformar preguntas realizadas en lenguaje natural a preguntas en lenguaje de consulta de una determinada base de datos, es decir, un software conversor de preguntas en lenguaje natural a *queries* de bases de datos. El objetivo del programa hacia el usuario es que con poca codificación éste pueda construir su propio sistema para acceder a partir del idioma natural a su base de datos. Si bien existen en el mercado productos similares que tienen una antigüedad de alrededor de 5 años (como Siri y Google Now) estos productos son de código cerrado. Quepy persiguió ubicarse en este nicho de productos pero constituyendo la única pieza aplicada de tipo *open source*.

- *Origen y fases de desarrollo del proyecto*

En el origen del proyecto confluyeron tanto una necesidad estratégica definida por la empresa, como el devenir de un proyecto anterior denominado SOSE (*Semantic Ontology Search Engine*).

[47] Entrevista a Patricio D.B., *Project Manager* de Proflow, 24/06/2014.

En 2012 la empresa definió una estrategia particular de negocios consistente en posicionarse en el mercado a través del desarrollo y posterior liberación de un producto *open source* puramente innovador. Así, surgió Quepy:

> *Por propósitos estratégicos, y la iniciativa de Marcos en su rol de CEO, para posicionar a la empresa, se decide hacer un 'proyecto innovador', para que los otros sepan que nosotros sí lo podemos hacer y nos quieran contratar; una estrategia publicitaria, digamos, con un plus adicional que viene por el lado de que a la gente de esta empresa le gusta hacer proyectos que entran en la categoría de open source. Entonces, a los empleados les gusta, da visibilidad fácil (los proyectos opens ource se difunden fácil en internet si son buenos). (…) vamos a hacer algo que sea innovador, y que nos de visibilidad, y que mantenga a la gente contenta*[48].

La apuesta por un producto *open source* se justificaba además en que éste ofrecía la posibilidad de exponer públicamente toda la mecánica interna desarrollada, de mostrar un grado elevado de calidad interna de los productos de la empresa, y además es de rápida y fácil visibilidad. Se buscaba un producto que oficiara como "carta de presentación" de las capacidades de desarrollo de la empresa:

> *El software open source es algo que puede verlo todo el mundo. Y no me refiero a la calidad del "cascarón" del producto, sino lo que está adentro. (...) Si fuera un reloj, y lo abrís, los engranajes tienen que ser dorados, tiene que estar precioso por dentro, porque esa es tu carta de presentación para la empresa que te quiere contratar* [49].

Con la misma motivación surgió el proyecto anterior, SOSE. Dicho proyecto, técnicamente, fracasó. Era un proyecto más ambicioso que Quepy, con la finalidad de desarrollar un software que permitiera el llenado de bases de datos desde lenguaje natural y que a la vez permitiera, a través de procesamiento de lenguaje natural, extraer y recomendar información desde esas bases de datos. Estos eran los objetivos "duros" del proyecto, a los cuales se sumaba un "objetivo suave" de posicionarse en la comunidad Python. Los objetivos duros no se lograron nunca con un nivel de calidad interna aceptable para la comunidad. La comunidad Python funcionó como testador y validador de los avances del proyecto. Durante el proceso de desarrollo de SOSE, miembros de la comunidad Python señalaron que dada la complejidad con la que se estaban manejando, uno de los objetivos intermedios del proyecto

[48] Entrevista a Rafael C., desarrollador de Quepy e impulsor de la idea del proyecto, 28/05/2014.
[49] Ibídem.

les estaba obstaculizando el desarrollo. Quepy surgió de ese punto, de la idea de extirpar ese objetivo intermedio, y hacer una pieza más simple y pequeña. De este modo nació Quepy como un proyecto *"más minimalista, menos ambicioso, pero mucho más concreto"*[50], que es una derivación de SOSE y del conocimiento preciso de los aspectos que no funcionaron allí.

El equipo de Quepy se conformó con integrantes del proyecto anterior. Un *Project Manager*, dos *developers* y una consultora desarrolladora. El *Project Manager* (Elías A.) que es quien gestionó el proyecto, fue analista funcional *part time* en SOSE y fue estudiante de Ingeniería en Computación (UNC). Entre los *developers* "puros" estuvieron Rafael C. (Licenciado en Ciencias de la Computación, FAMAF, UNC), docente especializado en *Natural Language Processing* con experiencia en programación de experimentos en este campo, y "John" (programador sin experiencia académica, pero con gran capacidad para comprender las ideas intuitivas que se le trasmiten y llevarlas a programación). El equipo de desarrollo se completaba con Laura A. (Dra. En Lingüística Computacional, Universitat de Barcelona), que reviste la figura de consultora-desarrolladora en el área de *Natural Language*, pero que no participó específicamente programando. Por último, hay un reconocimiento especial y simbólico dentro de la empresa a un integrante (Horacio D.) que participó de la idea en SOSE que dio origen al proyecto, pero que no programó específicamente en Quepy.

El desarrollo del proyecto puede dividirse en tres etapas: la primera tuvo lugar entre junio y octubre de 2012, la segunda etapa entre diciembre de 2012 y marzo de 2013, y la última etapa entre julio y septiembre de 2013.

La primera etapa se abocó al desarrollo de la primera versión de Quepy para generar consultas de base de datos en lenguaje SPARQL. La definición de utilizar este lenguaje por parte del equipo constituyó un hito en la primera semana del proyecto, que facilitó el resto del desarrollo e implicó capacitación y formación del equipo en el uso del mismo. Se trabajó en que la primera versión permitiera consultas en la base de datos de DBpedia (base que está formada por contenidos de Wikipedia).

Durante la segunda etapa se concretó un sitio demo de Quepy y se trabajó en la generación de documentación y en el agregado de la cantidad de preguntas reconocidas. Al interior de esta etapa los entrevistados identificaron dos hitos del proceso de desarrollo del proyecto.

[50] Ibídem.

Por un lado, a partir de la producción de la demo de Quepy se extendió esta práctica al resto de los proyectos y la empresa definió adoptarla como política de difusión y presentación. El otro hito fue la liberación del código de Quepy por versiones.

> *Llegamos hasta un hito y a partir de ese hito liberamos la versión 01 de Quepy (…) establecemos una línea base (…) lo primero que vamos a mostrar de Quepy llega hasta acá, y ahí mostramos. Luego hacemos más mejoras, cuando tenemos tiempo armamos un sprint y podemos sacar una versión 02. Y así vamos liberando el código de Quepy*[51]*.*

La última etapa del proyecto consistió en ampliar las *queries* de Quepy al lenguaje MQL, para poder generar consultas en una de las más grandes bases de datos que existen en la actualidad, Freebase[52]. Ello implicó capacitación y formación del equipo en el lenguaje MQL, y en la estructura y manejo de la base de datos Freebase.

- *Organización del proceso de trabajo*

Una de los aspectos en las que este proyecto resultó particularmente dinámico fue en las innovaciones en la organización del proceso de trabajo. Para el desarrollo de Quepy también se implementaron Metodologías Ágiles *Scrum* realizando adaptaciones acorde a una serie de factores.

Quepy funcionó a través de *sprints* de alrededor de 15 días. Al inicio y al final de cada *sprint* se realizaban reuniones de revisión y planificación donde se definía el *scope* o alcance del proyecto y del *sprint*, las metas, las prioridades, la planificación de las tareas necesarias para alcanzar cada meta, se estimaba y asignaba un tiempo de trabajo por tarea y se evaluaba la consecución de las distintas tareas. Así, el proceso productivo se organizó a través de una sucesión de *sprints*, donde quedaban congeladas las tareas correspondientes al período. Durante cada *sprint* se mantenían breves reuniones diarias de sincronización (máximo de 10 minutos) llamadas *standups*, donde se hablaba de qué se había hecho, qué se estaba haciendo y qué impedimentos habían aparecido. De haberlos, el *Project Manager* puede hacer algo al respecto y al mismo tiempo el resto del equipo queda notificado a partir de la reunión.

[51] Entrevista a Elías A., *Project Manager* de Quepy e impulsor de la idea del proyecto, 18/06/2014.

[52] Freebase es una base de datos colaborativa que compuesta por metadatos en línea obtenidos a partir de múltiples fuentes, incluyendo contribuciones "wiki" individuales, creada por Metaweb, que fue adquierda por Google en 2010. Los datos de Freebase están disponibles para uso free/libre, comercial y no comercial bajo una licencia *Creative Commons*.

El *Project Manager* explicó en forma estilizada la dinámica de los tres tipos de reuniones que se realizaban para cada *sprint*: una reunión inicial de *brain storming*, una reunión de *planning* y luego reuniones regulares y sucesivas de revisión. En la reunión inicial se trabaja sobre la definición del *scope* del proyecto y el problema a abordar, y se arma la primera idea del proyecto, buscando una idea genérica. En esta instancia tienen gran participación los desarrolladores y la consultora. En este tipo de reuniones aparecen lo que denominan *spikes*, que son "cosas para investigar" cuando no se pueden hacer estimaciones acerca de los tiempos de las tareas o acerca de cuáles son las tareas a realizar. Se plantea: *"Esto no lo puedo estimar, no sé si lo podríamos hacer así. Hay gran incertidumbre, entonces se dedica un tiempo para investigar esa línea"*[53], de lo que surgen tareas para indagar sobre esas cuestiones que cada integrante se lleva. Luego vuelven con los resultados de sus *spikes* a una reunión de planificación. En estas reuniones se busca concretar la identificación de tareas y una estimación de resultado e insumo-tiempo asociada a cada tarea, lo que permite planear una serie de iteraciones para alcanzar metas y ciertos "entregables". Se especifican también las tareas donde que no se pueden estimar y esto se tiene en cuenta para definir los *deadlines*. En este tipo de proyectos internos se pueden dejar algunas tareas sin estimar y crear tareas a mitad de una iteración e incluso redefinir su alcance, lo que se sale de un esquema de *Scrum* canónico. Por último, acompañando la sucesión de los *sprints* planificados, se realizan reuniones donde se evalúa lo que se hizo en cada iteración y se revisan las tareas para la siguiente. Se fija una fecha para el primer *release*[54]. Si se cumplió con todos los entregables establecidos para ese plazo, hay *release*, y en caso contrario se evalúan las razones del incumplimiento y se planifica nuevamente. El proceso continúa hasta alcanzar los entregables establecidos y determinadas metas de entregables finales que se especifican en las reuniones de *planning*, que establecen cierto umbral de calidad/resultados que constituyan un producto determinado satisfactorio para la empresa. La organización del trabajo a través de *sprints*, reuniones de *planning* y *standups* es propia de las Metodologías Ágiles que se implementan en la empresa.

En relación con los roles y funciones al interior del proyecto, se realizaron adaptaciones de los criterios clásicos de *Scrum*. En el caso de Quepy las figuras organizativas relevantes son: el *Project Manager*, dos *developers* y una consultora-desarrolladora. El rol de *Project*

[53] Entrevista a Elías A., *Project Manager* de Quepy e impulsor de la idea del proyecto, 18/06/2014.
[54] Es una versión de lanzamiento del producto.

Manager es análogo aquí al de *Scrum Master*[55] (Sutherland y Schwaber, 2012), de facilitador del equipo, quien evalúa cómo organizar los tiempos con los recursos con los que cuenta la empresa y da continuidad a las planificaciones. Gestiona la efectiva realización de todas las reuniones se hagan, el cumplimiento de los ciclos planteados, monitorea las agendas de los integrantes, establece el espacio para que los temas que traen los integrantes del equipo formen parte de las reuniones, documenta las reuniones y registra las tareas y estimaciones que surgen al sistema de gestión de proyectos de la empresa (es una herramienta de gestión de proyectos denominada JIRA[56]). Dentro del equipo de *developers* se practica la multifuncionalidad, sin distinciones entre funciones de desarrollo front-end, back-end, de codificación de arquitectura, mantenimiento de base de datos y sistema, testing, etc. para una misma persona. Las decisiones técnicas se toman en conjunto entre los desarrolladores y por consenso, y al ser un equipo pequeño desaparece la figura de *Technical Leader* propia de proyectos más grandes. La naturaleza de Quepy como proyecto interno de la empresa no dirigido a un cliente, hace desaparecer, además, la figura de *Product Owner* que prevén los estándares canónicos de *Scrum*. Es el propio equipo el que ejerce colectivamente ese rol junto con otros actores de la organización que van colaborando con ideas y acaban definiendo la arquitectura de requerimientos del proyecto, del software. De este modo, en el caso de Quepy los entrevistados identifican como *product owner* a los agentes que definieron cómo tenía que ser la herramienta: la consultora, el CEO y los desarrolladores que colaboraron con la idea (tanto los que formaron parte del equipo, como del proyecto anterior). En tanto, otros roles directamente fueron creados a partir de las necesidades y la experiencia de desarrollo del proyecto, como la figura de consultora-desarrolladora que asesoró en lingüística computacional en Quepy. Su participación era activa en las reuniones iniciales y de *planning*, y también se la tenía en cuenta para las reuniones de replanteo del proyecto o re-diseño del producto.

Otra de las principales adaptaciones, producto de la naturaleza interna del proyecto, es la consideración de tareas sin estimar dentro de las reuniones de *planning*. Es la aceptación de que, debido a que en un proyecto puramente innovador los resultados son altamente inciertos, es admisible dejar algunas tareas particulares sin estimar. También se permiten crear tareas especiales en medio del *sprint*, lo que no está previsto en un esquema de *Scrum* típico.

[55] A diferencia del Proflow, donde la figura de *Project manager* existe con la misma denominación pero asumiendo roles distintos, como se explicó más arriba.
[56] Es una herramienta de *management* de proyectos. Más información en https://www.atlassian.com/es/software/jira

Por último, realizaron adaptaciones operativas en relación con el grado de periodicidad de las reuniones de trabajo vinculadas a la disponibilidad horaria de los trabajadores.

- *Aprendizajes, desarrollo de capacidades y vinculaciones*

El proyecto partió de un conjunto de capacidades pre-existentes en el área de *Natural Language Processing* de dos de los integrantes del equipo. Además, contaban con el "funcionamiento comprobado" de todo el equipo en el proyecto anterior, aunque la mayoría de los integrantes tenía poca práctica en el proceso de desarrollo de software de la empresa (gestión de proyectos, idiosincrasia de la organización, etc.). Esto es, partían de un importante nivel de conocimiento de tipo codificado proveniente de su educación formal, pero menos *know how* en la producción de software.

Retrospectivamente, los entrevistados señalaron una variedad de aprendizajes derivados del proyecto, concentrados en gran medida en materia organizacional.

Uno de los primeros aprendizajes del proyecto (surgido en el primer *sprint*) fue el reconocimiento de la necesidad de gestionar los proyectos internos con el mismo nivel de profesionalidad que los proyectos para clientes, con adaptaciones particulares. La organización del proceso de trabajo en SOSE fue más rudimentaria que en Quepy, porque fue uno de los primeros proyectos internos, el proceso de las reuniones era desorganizado y utilizaban otro sistema de gestión de proyectos antes de JIRA. Durante el desarrollo de Quepy empezaron a aplicar formas de gestión de proyectos para clientes adaptados a las particularidades del proyecto interno y cambios en la empresa.

Uno de los aprendizajes más importantes que surgió de Quepy fue el reconocimiento de que este tipo de proyectos (internos) necesariamente implica tareas y logros que no se pueden estimar y que su desarrollo puede llevar una cantidad de tiempo incierta, y no converger necesariamente en un resultado útil. Esto es debido,

> *al hecho de que algunas de esas tareas tienen una altísima incertidumbre, es decir, en un proyecto de innovación decimos 'esto puede demorar esto', pero no sabemos realmente. Es algo innovador, y entonces no podemos estimar. Esto es una de las lecciones aprendidas que nos han dejado estos proyectos*[57].

[57] Entrevista a Elías A., *Project Manager* de Quepy e impulsor de la idea del proyecto 18/06/2014.

Y este hecho implicó un aprendizaje para todos los niveles de la organización: a los mandos altos y medios les permitió reconocer y aceptar este hecho, y a los desarrolladores no frustrarse cuando no se logran resultados de una tarea, dado que es una posibilidad inherente a la naturaleza del proyecto.

Otro de los aprendizajes relevantes del proyecto fue la importancia de realizar una demo para el avance del proceso de desarrollo. Esto generó una práctica que luego se expandió al resto de los proyectos de la empresa.

El desarrollo del proyecto en sí involucró además el desarrollo de competencias técnicas y la inversión en capacitación y formación del equipo en los lenguajes de bases de datos MQL, SPARQL y en la estructura de DBpedia y Freebase, que antes no se utilizaban.

Se destacan además importantes vinculaciones que les permitieron incrementar sus capacidades, funcionar con soporte en el proceso de desarrollo y contar con desarrollos de terceros como insumo. La comunidad Python concentra el grueso de las vinculaciones tecnológicas más relevantes para el proyecto, y en segunda medida las relaciones con la Universidad. La comunidad Python funcionó durante todo el proceso de desarrollo del proyecto anterior probando los distintos avances, y fue precisamente un miembro de esta comunidad quien identificó la posibilidad de desarrollar independientemente la pieza de SOSE que dio origen a Quepy. Además, en el proceso de desarrollo de Quepy se utilizaron librerías de código abierto realizadas en *Python* por terceros (por ejemplo, NLTK[58]), y al incorporarlas se hicieron consultas acerca de cómo implementarlas y se recibió soporte en ese sentido. Luego, la interacción continúa difundiendo lo que se ha hecho al momento de liberar el producto. La vinculación con la Universidad se dio directamente a través de quienes fueron *developers* del proyecto, Laura A. y Rafael C., que son profesores de FAMAF en la Licenciatura en Computación. Ellos han incorporado Quepy como herramienta de formación en la materia que dictan, y ésta ha sido incorporada para la realización de distintas tesis.

- *Resultados de innovación y medidas de performance.*

Un parámetro de *performance* del proyecto es la medida en que satisfizo la necesidad estratégica que le dio origen. En este sentido, Quepy cumplió el objetivo de posicionar y exponer a la empresa, que era su principal meta, tanto a través de la liberación del producto como a través de la generación de presentaciones en conferencias (no académicas) y

[58] Más información en: http://www.nltk.org/

premios en concursos. La aplicación ganó en 2012 una mención especial otorgada en el "**V Encuentro Nacional de Gobierno Electrónico" como parte d**el "Concurso de Datos Abiertos del Gobierno de Uruguay", en la categoría "ideas abiertas" de la presentación *Búsqueda por lenguaje natural en catálogo de datos abiertos*[59]. En 2014 aprovecharon el desarrollo de Quepy para participar del PyData Silicon Valley 2014 realizada en las oficinas de Facebook, donde presentaron la exposición *Querying your Database in Natural Language*[60].

La potencialidad que genera el proyecto terminado para participar de conferencias es uno de los principales resultados de innovación comercial para la empresa. Las presentaciones en conferencias y la liberación del producto son medios para difundir las capacidades de desarrollo con que cuenta la empresa, dado que toda la mecánica interna de la aplicación queda expuesta; y son también una oportunidad para el desarrollo de *networking* y acumulación de capital social con productores en sus áreas de *expertise*.

De este modo, se observa que el grueso de los resultados en materia de innovación del proyecto está en materia organizacional y en comercialización (posicionamiento de mercado).

El grado de novedad técnica de Quepy no radica en su aporte a la ciencia, sino que lo novedoso es la aplicación de un conocimiento existente a nuevos usos[61], su grado de simplicidad y su carácter *opensource*[62].

Machinalis no obtiene en la actualidad ingresos, ni por la venta de la aplicación en sí (que es de libre acceso) ni por servicios de asistencia técnica de soporte para el mismo, sino que los beneficios materiales del proyecto en sí tienden a ser más indirectos, poniendo en tela de juicio las aseveraciones típicas que relacionan la innovación radical con unos férreos derechos de propiedad intelectual.

[59] http://www.agesic.gub.uy/innovaportal/v/2436/1/agesic/tengo_una_idea:_entrega_de_reconocimientos_de_dateideauy_.html

[60] http://pydata.org/sv2014/abstracts/#197

[61] Rafael C., impulsor de la idea del proyecto y *developer* en el mismo, sostiene: *"la idea de resolver ese problema con esa técnica, es de los '60"* (Entrevista realizada el 28/05/2014), pero la técnica hoy se denomina distinto y el problema ha sido adaptado a los datos disponibles actualmente, esto es lo innovador.

[62] Respecto al producto completo, hay productos similares que tienen 5 años de antigüedad, como Wolfram Alpha, Siri (de Apple) y Google Now. Sin embargo, estos productos son de código cerrado y Quepy se ubica en este nicho pero, por su parte, es la única pieza aplicada, no puramente académica, que es *opensource* y está disponible libremente para resolver los mismos problemas, aunque un poco más básico.

4. Análisis Comparativo

El análisis comparativo de los dos tipos de proyectos permite hallar similitudes y diferencias entre los proyectos para clientes y los proyectos internos que caracterizan a una empresa innovadora del sector, como es Machinalis.

En primer lugar, si bien el proyecto para cliente resulta de un proceso típico de negocios en tanto que el proyecto interno resulta de un proyecto anterior y una estrategia específica de innovación de la empresa, en el origen de ambos aparece como central el carácter desafiante del proyecto tecnológico en sí. Esto es, el factor motivacional aparece como muy importante en el origen de los proyectos de una empresa innovadora. Por otro lado, por el momento de la historia de Machinalis en que ambos surgen (en sus inicios) parece natural el hecho de que compartan el objetivo de posicionamiento de mercado de la empresa.

En segundo lugar, otra de las similitudes más salientes entre los proyectos respecta al grado de especificación y plasticidad con que se estructura la organización del proceso de trabajo. En ambos se implementan Metodologías Ágiles *Scrum*, pero realizando adaptaciones acordes a i) la naturaleza del proyecto, ii) las características y cultura organizacional de la empresa, iii) las necesidades y características de los integrantes del equipo.

En tercer lugar, el desarrollo de competencias en ambos casos parte de un elevado umbral de competencias técnicas en las áreas de *expertise* de la empresa (*Natural Language Processing* en un caso, y *Complex Web Development* en el otro). De hecho, el personal de la empresa cuenta con un nivel de formación académica muy por encima de los niveles promedio del sector y en segmentos *top knowledge*. Por otro lado, buena parte de los aprendizajes se dan a través de procesos de *learning by failing* al enfrentarse a dificultades y de *learning by interacting* al participar de comunidades *open source* y vincularse con el cliente.

Así, en cuarto lugar, gran parte de las dificultades que se han ido sorteando durante el desarrollo de los proyectos han estado relacionadas con los modos de gestionar los tiempos de trabajo (cómo planificar tareas, estimar los tiempos de cada una, plazos de finalización de entregables, etc.) y el desarrollo de mecanismos efectivos de comunicación entre los integrantes del proyecto y con el cliente. Partiendo de una base importante de competencias técnicas, aparece con enorme relevancia para el caso el desarrollo de “competencias blandas”, lo que los entrevistados suelen denominar “*skills* sociales”, como la capacidad de

comprender y reelaborar necesidades de terceros, comunicar ideas y resultados, trabajar en equipo, etc. Este tipo de competencias operan de manera transversal en la actividad de producción de software y sirven de apoyo a las competencias técnicas específicas. De este modo, el grueso de los aprendizajes surgidos de estos proyectos se ha dado en materia organizacional y de gestión de proyectos.

En quinto lugar, se observa que la mayor parte de los resultados de los proyectos en materia de innovación se obtuvieron es el área organizacional, y en comercialización y posicionamiento de mercado en menor medida. El caso del proyecto interno muestra una particular medida de *performance* innovativa en materia de comercialización y marketing de la empresa, que es su capacidad para generar oportunidades de *networking* de negocios a través de las presentaciones en conferencias. A ello se suma la exposición pública mediante la liberación del producto y la obtención de premios en concursos.

En sexto lugar, aparece como muy relevante en ambos casos el grado de conectividad como fuente de desarrollo de competencias y de apoyo. De manera distintiva, en el caso del proyecto interno aparecen con mucha más fuerza las vinculaciones con la comunidad *open source*, mientras que en el caso del proyecto comercial las relaciones directas con el cliente tienen una relevancia primordial seguidas por las vinculaciones con instituciones de CyT y la comunidad *open source*.

Por último, cabe destacar que ambos proyectos presentan un nivel coincidente de innovación técnica, en tanto lo novedoso en este aspecto es la aplicación de conocimientos científico-tecnológicos existentes a un problema particular o un caso específico. Además, el grado de innovación técnica de los proyectos se materializa en aspectos no necesariamente visibles, mensurables o cuantificables, tales como innovaciones técnicas y mejoras en partes del producto y en desarrollos que se liberan en la comunidad y aportan al corrimiento de la frontera de posibilidades del *framework* que utilizan para desarrollar (Django). Finalmente, ambos proyectos constituyen innovaciones de producto, sea por tratarse de un software desarrollado a medida para un segmento del mercado que no registra antecedentes, o de un software que no se había desarrollado antes como producto *open source*.

5. Comentarios Finales

A lo largo de este trabajo hemos analizado en términos comparativos el desarrollo de dos proyectos tecnológicos en una empresa innovadora del sector de SSI de Córdoba, Argentina. En particular, hemos examinado su origen, las particularidades que asume la organización del trabajo, el papel de las competencias técnicas y organizacionales y las vinculaciones, desde una perspectiva del desarrollo de capacidades y la dinámica de la innovación.

En primera instancia, nos interesa destacar aquí las evidencias que presenta el caso en relación con la manera en que las actividades innovativas forman parte del propio proceso productivo de las actividades de desarrollo de software y provisión de servicios asociados. En particular, en los proyectos estudiados resaltan en este aspecto: el diseño de nuevos productos y procesos; I+D interna y externa; capacitación orientada a la innovación; implementación de estrategias de mejora continua; incorporación de software existente que implique mejoras para la empresa; incorporación de una consultora. En este sentido quedan de manifiesto las características diferenciales de la innovación en los sectores de servicios intensivos en conocimiento respecto a los sectores manufactureros tradicionales.

En segundo término, se observa que efectivamente las conductas creativas durante el desarrollo de los proyectos habilitaron la emergencia de innovaciones, principalmente organizacionales, que permitieron por un lado, superar las dificultades que las originaron, y por otro lado ampliar el alcance de las metas inicialmente planteadas en términos de innovación. Esto es: ante las dificultades, no se optó por conductas más adaptativas que implicarían el abandono o minimización de los objetivos de los proyectos originalmente impulsados, sino, en términos generales, por la implementación de estrategias de continuidad que dieron lugar a nuevas iniciativas de relevancia para la empresa en términos de innovación.

En tercer lugar, los aprendizajes organizacionales derivados de los proyectos permitieron a la empresa generar conocimientos que se materializaron en innovaciones que acabaron modificando prácticas internas al nivel de la empresa, que se extendieron a otros proyectos. En este caso, dado que no vuelve a producirse el mismo software (en serie, por ejemplo), la aplicación del nuevo conocimiento generado (la innovación organizacional) se realiza primordialmente en la producción de algún otro software o proceso que excede al originario: en otro u otros proyectos nuevos o en desarrollo. Este aspecto no deja de llamar la atención, por lo menos como una posible particularidad de los procesos de innovación en la actividad de software, en relación a los sectores manufactureros donde tradicionalmente las nuevas

prácticas organizacionales que surgen en el proceso productivo de un bien por lo general se incorporan a la producción del propio bien en cuestión, más allá de que la innovación organizacional pueda o no extenderse a otra gama de productos de la empresa.

Por último, ciertas particularidades de la firma expresadas en el desarrollo de los proyectos analizados movilizan interrogantes en torno a las condiciones para la innovación que involucran las herramientas y comunidades *open source*, a su vez en tensión permanente con los derechos de propiedad privada aun dentro de una misma unidad productiva y la necesidad de generación de ingresos. Ello tanto en términos de las innovaciones técnicas que habilita esta forma de producir, como en términos de los cambios organizacionales que presupone e impulsa, que pueden constituir fuentes novedosas de innovación en un sector con alto potencial de desarrollo económico atravesado por lógicas de producción diversas aunque no necesariamente contradictorias.

Referencias Bibliográficas

Ancori, B.; Bureth, A. y Cohendet, P. 2000. "The Economics of Knowledge: The Debate About Codification and Tacit Knowledge." *Industrial and Corporate Change*, 9(2), 255-87.

Antonelli, C. 2008. *Localised Technological Change: Towards the Economics of Complexity*. Routledge.

Cowan, R.; David, P. y Foray, D. 2000. "The Explicit Economics of Knowledge Codification and Taciteness." *Industrial and Corporate Change*, 9(2), 211-53.

Denzin, N. K. y Lincoln, Y. S. eds. 2005. *The Sage Handbook of Qualitative Research*. Sage.

Drejer. 2004. "Identifying Innovation in Surveys of Services:A Schumpeterian Perspective." *Research Policy*, 33(3), 551-62.

Eisenhardt, K. M. 1989. "Building Theories from Case Study Research." *Academy of management review*, 14(4), 532-50.

Ernst, D. y Lundvall, B.-Å. 2004. "Information Technology in the Learning Economy: Challenges for Developing Countries," E. Reinert, *Globalization, Economic Development and Inequality.* GB: Edward Elgar Publishing, 258.

Gallouj, F. y Savona, M. 2009. "Innovation in Services: A Review of the Debate and a Research Agenda." *Journal of Evolutionary Economics*, 19(2), 149-72.

Gallouj, F. y Weinstein, O. 1997. "Innovation in Services." *Research Policy*, 26(4), 537-56.

Lundvall, B.-A. 1996. "The Social Dimension of the Learning Economy," DRUID Woking Paper Nº 96-01, Aalboarg.,

Lundvall, B.-ä. y Johnson, B. 1994. "The Learning Economy." *Journal of industry studies*, 1(2), 23-42.

Lundvall, B. Å. ed. 1992. *National Systems of Innovation: Towards a Theory of Innovation and Interactive Learning*. London: Printer Ed.

Miles, I. 2005. "Knowledge Intensive Business Services: Prospects and Policies." *Foresight*, 7(6), 39-63.

Muller, E. y Doloreux, D. 2009. "What We Should Know About Knowledge-Intensive Business Services (Kibs)." *Technology in Society*, 31(1), 64-72.

Neergaard, H. 2007. "Sampling in Entrepreneurial Settings," H. Neergaard y J. P. Ulhøi, *Handbook of Qualitative Research Methods in Entrepreneurship.* 253.

Neergaard, H. y Ulhøi, J. P. eds. 2007. *Handbook of Qualitative Research Methods in Entrepreneurship*. Edward Elgar Publishing.

Nelson, R. y Winter, S. 1982. *An Evolutionary Theory of Economic Change*. Harvard University Press.

Niosi, J.; Athreye, S. y Tschang, T. 2012. "The Global Computer Software Sector." *Economic Development As a Learning Process: Variation Across Sectoral Systems*.

Nonaka, I. 1995. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation: How Japanese Companies Create the Dynamics of Innovation*. Oxford university press.

Schumpeter, J. A. 1947. "The Creative Response in Economic History." *The journal of economic history*, 7(2), 149-59.

Sutherland, J. y Schwaber, K. 2012. "The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework " Cambridge, MA: Scrum Inc.,

Takeuchi, H. y Nonaka, I. 1986. "The New New Product Development Game." *Harvard business review*, 64(1), 137-46.

Yin, R. K. 2009. *Case Study Research: Design and Methods*. Sage.