"Un nuevo Patrón de Diseño de Software: Programmer's Attitude"

Guillermo G. Pantaleo
Departamento de Computación
Facultad de Ingeniería, Universidad de Buenos Aires, Argentina.
Correo Electrónico: gpantaleo@fi.uba.ar

Daniel A. Roitbarg
Logismata SA, Buenos Aires, Argentina.
Correo Electrónico: droitbarg@yahoo.com

Introducción

Mucho se ha dicho acerca de las relaciones entre las organizaciones que desarrollan software, la calidad de los productos por ellas generados y la arquitectura del software de dichos productos. También se ha escrito acerca de cómo estas relaciones afectan a dichas arquitecturas [8, 9, 10]. Sin embargo, poco se ha dicho de las características de las personas que forman parte de un grupo de desarrollo de software, y de como se relacionan las emergencias de ciertas actitudes personales de algunos de ellos con las características de la organización donde desarrollan su trabajo y de los proyectos de software en los cuales participan. Este documento trata el problema generado por estas actitudes y propone una solución en el formato de Patrón de Diseño de Software. La presentación esta basada en el formato de los "patterns writters" [ref. 1]. Se siguen las pautas acerca del descubrimiento de nuevos patrones. El problema que trata de resolver el patrón presentado fue vivido por los autores a lo largo de años de experiencia en distintos proyectos de software.

Desarrollo

Categoría de Patrón: Imprescindible.

Categoría de Problema: Factor humano.

Nombre : Programmer's Attitude.

Contexto : Frente a todo proyecto de software, las distintas actividades a llevar a cabo antes de su inicio y por parte de los responsables comprende la elaboración de un plan en el cual se esbozan lineamientos básicos como a) la categoría del sistema a desarrollar(Web, embebido, de oficina, etc.), b) la metodología a utilizar en el proceso de desarrollo (Proceso Unificado de Desarrollo, Extreme Programming, etc.) que se seguirá para arribar al producto final, c) el tipo y cantidad aproximada de desarrolladores (analistas, diseñadores, programadores, administradores de base de datos, testeadores). A partir de allí se recorre el camino (proceso de desarrollo [ref. 3], [ref. 7]) hacia la obtención del sistema. Otra posibilidad, muy vista por estos días es la contratación de desarrolladores, la elección de algunas herramientas de desarrollo y el comienzo del desarrollo sin ninguna metodología para el proceso de desarrollo. Sí, aunque es difícil de entender, todavía (año 2002), se conforman grupos de desarrollo que se hacen a la aventura de implementar un sistema sin proceso ordenado ninguno. En ambos casos, existen desarrolladores, seres humanos con virtudes, defectos y características individuales que los hace distintos unos a otros [ref. 2]. Los hay capaces, inteligentes, creativos, trabajadores, habilidosos, regulares, mediocres, haraganes y es posible que hasta incapaces. Sin embargo todos tienen una característica común que es la clave del

problema a resolver. Esa característica se llama "actitud". Esta, es fundamental y constituye el talón de Aquiles o la Estrella Guía de todo emprendimiento humano, en particular, de todo proyecto de software. Si bien es independiente del tamaño del proyecto, es más notoria en proyectos que involucran grupos humanos medianos y grandes.

Problema : se plantea el problema a partir de las fuerzas que participan de él.

Fuerzas

1. ***Programadores "seniors" de personalidad egoísta.***

   Existen en el ambiente de la informática, son programadores con muy buena formación y experiencia. Son el tipo de recursos humanos que todo líder de proyecto desea tener en su grupo, por su capacidad de creación y trabajo. Sin embargo poseen una actitud egoísta frente a sus colegas. No están dispuestos a compartir su conocimiento. Lo esconden, lo retacean. Son mezquinos, no poseen la humildad de los que "saben". Este comportamiento genera conflictos y en definitiva conspira contra el funcionamiento del grupo. A veces obedece a personalidades inseguras, otras simplemente a la falta de comprensión del significado del conocimiento y la relación de este con la evolución humana. Frente a esta actitud es a veces preferible no contar con este tipo de desarrolladores.

2. ***Programadores " juniors" que se creen genios.***

   En los últimos anos ha habido una explosión de nuevos lenguajes y tecnologías, que hacen difícil un conocimiento detallado por parte de los desarrolladores. Sin embargo, en toda disciplina y quizás mas en esta que en otras, la experiencia constituye un ingrediente fundamental. Es un escenario repetido encontrar jóvenes programadores que al conocer algún lenguaje, alguna tecnología novedosa, se sienten genios. Esta actitud cuando es acompañada de una visión que menosprecia la experiencia, es otra de las amenazas de conflicto dentro de un grupo de desarrollo; y la consiguiente posibilidad de fracaso del proyecto.

3. ***Líderes de proyecto con mala formación y sin experiencia***

   Con el advenimiento del "boom" Internet, el fenómeno globalizador y el avance tecnológico se asentaron en países en desarrollo empresas multinacionales; de telecomunicaciones, bancos, de distribución de energía eléctrica, etc. De esta forma se genero empleo en el área de informática, en todas sus especialidades. Desde el gerenciamiento de unidades de negocio pasando por el planeamiento estratégico, marketing, ventas, atención al cliente y desarrollo. Se creo la cultura del ejecutivo de cuentas. Un escenario muchas veces visto fue el descuido del área de desarrollo. Las gerencias y jefaturas de proyecto de dichas áreas fueron puestas en manos de personas que si bien poseían formación técnica, esta no era actualizada o no tenían suficiente experiencia. Cuando se repite este escenario, lo que ocurre es que se

establece un canal de comunicación entre los desarrolladores y los responsables de los proyectos que en general no funciona, ya que hablan distintos idiomas. No existen cargos jerárquicos en el área de desarrollo en países en desarrollo, como sí los hay en países desarrollados. Las actitudes generadas por los responsables de los proyectos a raíz de su mala comunicación con el resto de los miembros del grupo de desarrollo y debida a la estructuración de las empresas, conspira contra el éxito de todos los proyectos de desarrollo de las empresas..

4. Cargos y no roles.

Las metodologías a seguir en el desarrollo de software han evolucionado a lo largo de los años. El Diseño Estructurado de Sistemas fue superado por el Proceso Unificado de Desarrollo. La tendencia que prevalece es que los miembros del grupo de trabajo ya no ocupen cargos (analista, diseñador, programador, testeador), sino que ocupen roles. Todos los desarrolladores poseen mayor conocimiento y experiencia en alguna de las tareas que se realizan en las distintas etapas del proceso de desarrollo, de allí el rol que juegan en cada una de ellas. Pero todos participan en todas las tareas con distintos roles. El paradigma de programación orientada a objetos acortó las distancias entre los requerimientos y la implementación, derribando los limites que imponían otros paradigmas entre las diferentes tareas de desarrollo. El proceso de desarrollo es ahora dinámico, para lo cual son necesarios desarrolladores que conozcan el paradigma y puedan jugar distintos roles. La actitud de miembros del grupo de desarrollo que no lo entiendan así, y persistan en ocupar cargos y no jugar roles, conspira contra el buen funcionamiento del grupo.

5. Héroes

Los héroes son habituales en determinados proyectos de software. Los héroes no son exclusivos de los proyectos de software, pero abundan en estos por la gran cantidad de proyectos con practicas ad-hoc, estimaciones poco creíbles y/o objetivos altamente maleables y volátiles que abundan en este medio. El héroe tiende a aceptar responsabilidades que lo exceden y que no están claramente distribuidas entre los miembros del grupo. Las personas con estas características tienden a cargarse de responsabilidades cuando sienten que el proyecto en el que están trabajando puede naufragar. Puede suceder que otros miembros del grupo se encuentren cómodos con el rol de héroe ya que de esta forma se verán relevados de responsabilidades asumidas por éste. Un caso especial se da cuando el héroe es el líder del equipo. Este genera entre los programadores la sensación de estar siempre en falta y que su esfuerzo nunca es suficiente. Por lo general esta actitud la asume alguien que es adicto al trabajo.

6. Puristas y exquisitos contra prácticos

Muchos estudiantes de matemática, en la universidad, opinan que la física es estéticamente fea. Lo que más los indigna son las aproximaciones, las cuales son

vistas como matemáticamente “desagradables” y “desprolijas”. Matemáticos y físicos tienen distintos objetivos. En los primeros se intenta establecer un corpus lógico cerrado riguroso y estético. En los últimos la matemática es solo una herramienta para poder describir los fenómenos naturales. En el ámbito del software las cosas no son tan claras. Sin embargo, existen quienes en nombre del buen diseño no escriben una línea de código si el código no se ajusta a “el mejor patrón de diseño” que corresponde al caso. Un ejemplo es el profesor de la universidad que dictaba el curso de programación orientada a objetos y que estaba totalmente en contra de la herencia múltiple. Su opinión era que si se daba una situación en la cual la herencia múltiple aplicaba, siempre había un diseño mejor sin herencia múltiple que modelaba el problema. No es esta afirmación un hecho alarmante en sí mismo. No hay duda que existe múltiples diseños para modelar el mismo problema. Lo alarmante es el hecho de negarse a usar por ejemplo la herencia múltiple en los casos en que seria sencillo y tal vez sensato hacerlo por circunstancias concretas; y solo aceptar refactorizar el sistema para evitarlo, como afirmación categórica, sin tomar en cuenta las necesidades del proyecto específicamente. Los purismos y fanatismos, justificables o no, hacen perder del foco principal al proyecto de software para concentrarse específicamente en el código [Ver intereses disímiles... mas abajo]. No es extraño por eso que existan proyectos de software exitosos desarrollados por especialistas en el dominio y no en software.
El caso opuesto es también peligroso. Existen desarrolladores que en nombre de la practicidad justifican cualquier forma de escribir software. Esto lleva por lo general a sistemas de difícil o casi imposible mantenimiento. Estas actitudes constituyen un obstáculo cuando se ponen de manifiesto en el transcurso del desarrollo de cualquier sistema.

7. Intereses Disímiles sin foco en el producto

Existen integrantes de grupo que anteponen sus intereses personales por sobre el objetivo fundamental que es la construcción de un producto de software. En muchas ocasiones algunos desarrolladores están mas preocupados en el código y en las innovaciones que pueden probar que en el producto mismo. Esto es algo que sucede comúnmente. Las carencias en la comunicación de las organizaciones para transmitir una visión globalizadora del producto software que se desea desarrollar y la falta de conformación de grupos consolidados de trabajo con objetivos comunes, fomentan este comportamiento. Un desarrollador aislado de la empresa en la cual trabaja y que no puede proyectarse en ella o que no se identifica con el producto que hace, tendera a escribir código de acuerdo a los criterios que mas se acerquen a sus objetivos personales, ajenos a su trabajo, y probar nuevas tecnologías que puedan impulsar su carrera en el futuro, obviamente, fuera de la empresa en que trabaja. Esta actitud al emerger durante el desarrollo degrada el proceso y constituye un obstáculo más.

8. Gerentes “Techies”

Quieren probar toda la tecnología nueva en el desarrollo. Generan cambios adhoc solo para satisfacer su curiosidad y necesidad de conocimientos, generando

frustración en el grupo de trabajo. La curiosidad por la tecnología es tal vez el motor principal que nos apasiona a los que estamos en este rubro. Suele suceder que los desarrolladores no ponen el foco en el producto que construyen como unidad mínima y básica, sino en las pequeñas partes del código en donde pueden probar las distintas nuevas tecnologías o alguna solución original a un problema. Cuando un gerente de desarrollo tiene estas características el proyecto esta en problemas. Es importante que el gerente no anteponga su curiosidad a las necesidades reales del proyecto. La rápida y a veces efímera tecnología cambiante genera la sensación que casi cualquier desarrollo de cierta envergadura (que tenga mas de 6 meses de empezado) parezca tecnológicamente "obsoleto". Nuevas tendencias y modas aparecen en el mercado a un ritmo vertiginoso y nuevas versiones mejoradas de productos, frameworks, metodologías, herramientas son constantemente ofrecidos. Este hecho objetivo combinado con un gerente "Techie", o fanático de la tecnología compulsivo, pueden fácilmente desviar el objetivo del desarrollo. En estos casos el producto software que satisfaga las necesidades de los usuarios se pierde de vista. Rápidamente, la sensación de estar perdiendo el tren de la tecnología se apodera del gerente y es probable que disponga de esfuerzos importantes para hacer actualizaciones de herramientas o cambiar mecanismos que funcionan por otros más nuevos y más "cools". Esta actitud es particularmente nociva, al igual que todas las que afectan a los responsables de la conducción del proceso de desarrollo.

9. Gerentes que no entienden lo importante

El factor humano en el desarrollo de software es determinante. Los intentos de formalizar procesos de software como procesos industriales (líneas de producción) o como negocios de venta al público pueden generar daños al grupo de desarrollo de distintas e importantes magnitudes. Algunos gerentes asumen que los desarrolladores son piezas intercambiables en el proceso de desarrollo. No valorar al desarrollador como una pieza clave en el desarrollo lleva a altos niveles de recambio, entre otros problemas. Muy pocos gerentes tienen en cuenta el costo de recambio(turnover) que suele ser muy alto. Esto sin tomar en cuenta el efecto de la desmotivación y su consiguiente baja en la productividad [ref. 11]. La adición de personas a los proyectos no genera un aumento lineal en la productividad. Además del costo de recambio (turnover) que implica la capacitación de la persona nueva en el sistema, recursos productivos pierden productividad para dar soporte a los recursos nuevos. Esto además de que los canales de comunicación se incrementan de acuerdo a la formula n(n-1)/2, agregar personas a un desarrollo puede ser mas una complicación que un aumento en la productividad si no se controla bien el proceso.

Reducciones de costos a través de espacios físicos pequeños y poco adecuados para el desarrollo producen también grandes mermas en la productividad además de frustración que aumenta las posibilidades de "turnover" [ref. 11].

La perspectiva del desarrollo de software como un negocio de atención al público hace que los desarrolladores generen como protección actitudes altamente nocivas. Estas prácticas se manifiestan por lo general con la imposición de normas sobre vestimenta y apariencia personal, como así también por horarios rígidos o conteos de horas trabajadas. El mensaje es recibido claramente como que la apariencia es más

importante que el trabajo real. De esta forma, se vuelve más importante para el desarrollador que quiere mantener su trabajo, ser visto en la empresa en las horas que el manager está presente y vestir elegantemente. Como la característica del desarrollo de software hace que la tarea de producción sea de dificultosa medición, es muy probable que un desarrollador que trabaje en una empresa con estas características produzca poco, se preocupe poco en la producción, haga horas extra innecesarias (incluyendo a veces fines de semana) solo para aparentar cumplir con los requisitos de la organización. Esta actitud es característica de lideres faltos de experiencia en la conducción de un proceso de desarrollo.

10. Fundamentalismo Informático

Los desarrolladores de software no están excluidos de otros fenómenos que ocurren a los seres humanos. De hecho, los desarrolladores de software **son seres humanos**. Los influjos a las tendencias y modas hacen de los desarrolladores un interesante mercado para vender productos llamados "metodologías" o lenguajes de programación y toda las gamas de productos que las soportan. Esto no es un mal en si mismo si no fuera que para poder vender estos productos se usan ideas de marketing que influyen seriamente en los desarrolladores poco experimentados o con visiones simplistas del proceso de desarrollo de software. Las visiones de "One size fit all" o "teoría unificada de las metodologías de software" no persiguen objetivos filosóficos o estéticos como en la física moderna. Ni siquiera persiguen objetivos prácticos, sino comerciales. Asimismo estas tendencias cuyas ideas de base son muchas veces buenas para determinados desarrollos se propagan como ideas religiosas y apuntan a ingenuos desarrolladores que desean experimentar el placer del mantra metodológico o el nirvana organizacional para tapar angustias personales y laborales.

La informática y en especial la programación de computadoras ofrece la oportunidad de adquirir ciertas habilidades que permiten generar trabajo productivo visible a corto plazo. Existen infinidad de formas de aprender ciertas habilidades que hacen de una persona un programador productivo. Pero el conocimiento adquirido de esta forma, carece muchas veces de contexto cronológico, de método de aprendizaje sistemático que establezca relaciones entre distintos aspectos de los diferentes temas, en contraste con una formación académica ordenada e inteligente. Existen entre muchos de los programadores surgidos así, tendencias al estereotipo "del fanático informático". A veces el fanatismo se manifiesta a través de un determinado lenguaje de programación. Este tipo de desarrolladores no es consciente que la selección del lenguaje de programación es una de las decisiones ha tomar durante el proceso de desarrollo. Esto no es visto así, ya que debido al tipo de formación adquirida se desconocen las metodologías para los procesos de desarrollo o se es fanático de UNA en particular.

Los fanatismos no solo afectan a los grupos de desarrolladores que carecen de formación académica. De hecho existe mucha literatura de proyectos devenidos en fracaso donde se aplicaron estrictas y pesadas metodologías dirigidos por profesionales con educación superior. Muchas veces, el exceso de formalismo hace que se implementen muchos procesos de dudoso beneficio que exigen arduo trabajo a los desarrolladores. Esto influye en forma directa en el animo y productividad del desarrollador, que en vez de concentrarse en lo que le gusta y sabe hacer, debe

también documentar y ocuparse de tareas administrativas que sobrecargan su trabajo. La documentación producida no siempre es útil (muchas veces es inútil) e implica una carga gigante en el proceso de modificación o cambio de requisitos para la readaptación del software. Esto vuelve al trabajo rutinario, rígido y por sobre todo aburrido

Como contraposición a estos procesos, aparecieron las metodologías ágiles. Estas ofrecen liberar al desarrollador de la pesada carga de procesos y documentación inútiles y los concentra en el éxtasis de escribir código casi exclusivamente. El problema es cuando una metodología ágil no prescribe ni se adapta a las necesidades del proyecto. La metodología ágil puede también ser ineficiente y no ser adecuada. Habrá que esperar la próxima moda para encontrar la solución

Las "sectas" que idolatran tal o cual lenguaje o metodología, ignorando que estos son solo herramientas, pueden estar formadas por grupos de personas heterogéneas con formaciones disímiles. Como se cuestiona en [ref. 5], estos son aspectos accidentales y no esenciales de la programación de computadoras. La actitud de programadores de esta categoría conspira también contra el éxito de todo proyecto.

Solución:

Estructura estática

La solución al problema planteado se obtiene a partir de entender y tomar decisiones precisas en la siguiente dirección: el desarrollo de software es una actividad realizada por seres humanos, por lo tanto debe considerarse la "actitud" humana frente al grupo de desarrollo y al proyecto en su totalidad, como uno de los temas fundamentales en el planeamiento. **La clave de la solución esta en el "grupo de trabajo", en su constitución, en las reglas que norman su funcionamiento y en la conducción del proceso de desarrollo seleccionado. El trabajo en grupo con reglas acordadas en forma explicita es el ambiente adecuado para evitar que emerjan las actitudes enumeradas como fuerzas del problema.** Aunque parezca trivial, el grupo de trabajo es la solución. Sin embargo la formación y conducción de un grupo humano no es sumar gente y repartir tareas. Esto trata el resto de esta sección.

Estructura de las empresas: las empresas deben jerarquizar a los desarrolladores, estos son los únicos capaces de gerenciar un área de desarrollo.

Las características problema que describimos denotan comportamientos inherente al comportamiento humano, por esta razón al potenciarse se vuelven nocivas dentro de un marco social. Y este marco social, no es mas que el grupo de trabajo, o el equipo de desarrollo de software. La aparición de estos comportamientos demuestran, en la mayoría de los casos, problemas en la organización y falta de conformación de grupos consolidados de trabajo. Estos son síntomas que deben ser tenidos en cuenta ya que su aparición denota grave peligro de fracaso del proyecto de software. La atenuación de los riesgos que hacen peligrar un proyecto de software se debe en gran medida, aunque no únicamente, a la conformación y consolidación del grupo de trabajo y a su relación y comunicación con la gerencia media y alta. Estos tres factores están relacionados

estrechamente. Por un lado las actitudes mencionadas, que conspiran contra el grupo de trabajo y por otro gerencias medias y altas no cumpliendo bien sus roles. Una gerencia media que no entienda la necesidad de un equipo de trabajo consolidado y que confíe plenamente en este, jamás podrá formar tal equipo; una gerencia alta que no pueda identificar claramente los objetivos de la empresa difícilmente de lugar a una gerencia media que pueda generar un equipo consolidado de trabajo. El desarrollo de software a nivel empresarial es sin duda una tarea compleja que requiere la coordinación y la suma de talentos de varios actores con distintas características. Estos procesos requieren como requisitos fundamentales los siguientes:

1. **Identificación de Objetivos del proyecto de software por parte de la Organización**
2. **Comprensión de la alta gerencia de los procesos de desarrollo de Software**
3. **Gerencia media Idónea en el ámbito técnico y gerencial.**

1 . Identificación de Objetivos del proyecto de software por parte de la Organización

Organizaciones que no tienen en claro sus objetivos estratégicos con el desarrollo de un producto software, difícilmente podrán generar las condiciones adecuadas para generar un producto. Esto sucederá a pesar de contar con el mejor grupo humano disponible.

2 . Comprensión global de la alta gerencia de los procesos de desarrollo de Software

El proceso de desarrollo de software es muy distinto a otro tipo de desarrollo de ingeniería. Esto se debe en gran medida a que lo que se esta desarrollando es conocimiento. Se va conociendo a medida que se va haciendo. Esto hace de los desarrolladores de software trabajadores únicos con poco paralelismo con trabajadores en otros rubros. Es necesario que ese desarrollador tenga el conocimiento y capacidad de resolver por si mismo problemas que requieren de talento. Y el trabajo en equipo se vuelve determinante. La información es la estrella y la comunicación fluida facilita su intercambio. También a la hora de tener que actuar ante situaciones de compleja solución o de presión. La tendencia de muchas organizaciones que no comprenden esto, es tratar a sus desarrolladores como piezas intercambiables. Esto conspira contra sus propios objetivos y son causa de múltiples fracasos dentro de lo que se conoce como el "problema del software". Es importante remarcar que no se esta diciendo que la alta gerencia deba ser especialista en software sino que debe de comprender las características únicas y particulares de los procesos de software y sus actores, para poder brindar el soporte institucional y material para recorrer el proceso con el menor riesgo.

3 . Gerencia media idónea en el ámbito técnico y gerencial

Es sin lugar a dudas la gerencia media la que carga con la mayor responsabilidad en un proyecto software. La gerencia media debe ser idónea técnicamente. Pero también debe ser capaz de generar un equipo consolidado de trabajo, distinguir quienes de los desarrolladores es mejor en cada área. Asimismo debe coordinar y dar al grupo los objetivos comunes necesarios para su conformación y mantenimiento como grupo

consolidado. Asimismo debe de actuar de nexo entre la gerencia alta y el grupo de desarrolladores manteniendo y generando canales de comunicación fluidos. También debe ser capaz de gestionar en pos de las necesidades de los desarrolladores como también capaz de sostener y aislar a estos de problemas que son ajenos a aspectos técnicos.

Proceso de desarrollo: Debido a la complejidad creciente de los requerimientos, es absolutamente necesario un proceso de desarrollo que se ajuste a las necesidades del tipo de proyecto. Este proceso debe ser conocido por todos los miembros del grupo de desarrollo y todos deben de estar convencidos que es el mejor. No es nuestro propósito entrar en detalle acerca de las distintas metodologías. Lo importante es remarcar que la metodología proporciona un lenguaje común y un contrato formal entre los distintos actores.

Líder del proyecto: el proyecto debe estar en manos de un profesional con conocimientos y experiencia no solo en tecnología, sino también en comportamiento humano.

Grupo de desarrollo: se debe tratar de evitar escenarios como los enumerados en la sección Problema. Para esto es muy importante fomentar el funcionamiento del grupo de desarrollo como un cuerpo único. Esto cambia las actitudes enumeradas por cooperación y solidaridad. Es importante entender que el éxito compartido es muy difícil de alcanzar, pero infinitamente más gratificante que el éxito personal. La experiencia recogida de un proyecto exitoso realizado en grupo y jugando distintos roles, es el antecedente más valioso que un desarrollador puede presentar en su curriculum. **La clave de la solución al problema esta en la constitución del "grupo de trabajo"**. Este grupo no es la suma de sus miembros, tiene identidad propia. En el grupo cada miembro cumple un rol precisamente determinado en cada etapa del proceso de desarrollo. Esta pequeñina sociedad contiene a cada uno de los miembros. En esta sociedad los miembros tienen derechos y deberes, establecidos en forma explicita cuando se les explica las pautas y normas de funcionamiento del grupo. Esta es una sociedad con reglas para que cada uno conozca la forma de relacionarse con el resto; y sepa que debe dar y que debe esperar. De esta sociedad todos forman parte y pueden recurrir a ella cada vez que lo necesiten. Esta sociedad por el solo hecho de estar formada por todos los miembros, esta por encima de cualquiera de las actitudes individuales. En particular de las mencionadas en la sección Problema. En este contexto, priman las actitudes conformes con las pautas acordadas para el funcionamiento del grupo de trabajo. Las conductas emergentes nocivas que analizamos anteriormente son contrarrestadas por las reglas mencionadas. Estas reglas son preestablecidas de antemano por los conductores del proceso de desarrollo; aunque algunas de ellas son acordadas por consenso con los miembros del grupo. Todas las "actitudes" analizadas que constituyen el problema a resolver, tienen razones diferentes. Sin embargo todas ellas tienen una alta probabilidad de emergencia en un contexto con carencia de metas claras, reglas de funcionamiento claras y confusión de roles. Estas son las características comunes a organizaciones de desarrollo de software sin grupos de desarrollo bien constituidos, sin objetivos concretos y sin

procesos de desarrollo. En escenarios como estos emergen estas "actitudes" que en ultima instancia se pueden explicar como distintos comportamientos frente a la lucha por la vida. Ocurre lo mismo en las sociedades sin organización, el caos imperante hace que emerjan las "actitudes" transgresoras e individualistas.
Respecto de la conformación del grupo de trabajo, no existen pasos específicos dados en un orden predeterminado para generar un grupo de trabajo consolidado. El tema esta estudiado y es sabido que los grupos pasan por diferentes etapas hasta lo que llamamos consolidación. Estas etapas son Formación, Tormenta, Norma, Actuación y Transformación [1] [ref. 12]. La importancia de los grupos de trabajo consolidados radica en que su eficiencia es muy superior a los grupos de trabajo que son solo la suma de talento individual [ref. 13]. También existen mediciones sobre la eficiencia de los desarrolladores comparadas y existen pruebas que un buen desarrollador puede ser un orden magnitud más eficiente que uno malo. Por eso es sumamente importante que los integrantes del grupo de trabajo sean talentosos. La gerencia media es responsable de gerenciar y liderar a los grupos de trabajo a lo largo de las etapas que llevan a la consolidación.
Entre los aspectos importantes que debe manejar la gerencia media en el proceso formación y consolidación de equipos de trabajo, mencionamos:

- Objetivos claros.
- Previsibilidad
- Metodología.
- Comunicación fluida con la gerencia, especialmente cuando hay decisiones que afectan la calidad del trabajo
- Participación del grupo en las estimaciones.
- Walkthroughs, Revisiones de código
- Participación en decisiones de diseño.
- Actividades conjuntas fuera del trabajo, simposios internos, capacitación continua.

A pesar de no haber reglas fijas para consolidar un grupo de desarrollo, es claro identificar algunas situaciones que conspiran contra su formación. Entre ellas podemos citar:

- Controles persecutorios. Horarios y tareas no importantes. Vestimenta.
- Falta de confianza en los miembros del grupo de trabajo.
- Intromisión de la gerencia en la tarea de los desarrolladores.
- Objetivos cuyos tiempos son calculados ad hoc, sin consultar a los miembros del grupo.
- Falta de Objetivo común del grupo de desarrollo.
- Falta de hitos a lo largo del desarrollo.
- Instalaciones inadecuadas. Lugar físico deficiente.
- Separación física de los miembros del grupo.
- Relajamiento en la calidad del producto generado, ya que los desarrolladores ven afectado su prestigio profesional.

[1] Los nombres originales en ingles son Forming, Storming, Norming, Performing y Transforming

Los argumentos expuestos y el análisis realizado de los grupos de trabajo, como solución a las actitudes que originaron nuestro problema, son presentados en este trabajo como ventajas. Sin embargo los autores no ignoran que al formar un grupo de desarrolladores para compartir las distintas actividades que involucra el proceso de desarrollo, siempre existen conflictos como en todo grupo humano. Estos pueden ser conflictos personales entre miembros del grupo o conflictos entre alguno de los miembros y el resto del grupo. Existe vasta bibliografía sobre técnicas y mecanismos para resolución de conflictos, sin embargo este tema esta fuera del alcance de este trabajo [ref.. 14].

Comportamiento

La dinámica de la solución propuesta esta basada en un **canal de comunicación fluido** entre los gerentes y los responsables del proyecto; entre los responsables del proyecto y los desarrolladores; y entre todos los desarrolladores entre sí. El grupo de desarrollo lo forman todos, el éxito o fracaso será compartido. Los desarrolladores al trabajar en una empresa que jerarquiza su actividad se sienten motivados y son más productivos. Los desarrolladores al estar convencidos del proceso de desarrollo que se utiliza, trabajan motivados y generan mejores productos. Deben fomentarse reuniones de actualización y cursos de capacitación con la participación activa de todos los desarrolladores. Estas reuniones deben ser en horario de trabajo y no representar una carga horaria adicional. La empresa debe estar convencida que esta realizando una capitalización y no un gasto. La discusión en este ámbito mejora la comunicación y evita las relaciones enfermas generadas por las actitudes mencionadas en la sección Problema. Debido a que el desarrollo de software es fundamentalmente una tarea grupal, es imprescindible fomentar todas aquellas actividades que contribuyan a la consolidación del grupo de trabajo. En este sentido, se trata de lograr un funcionamiento coordinado, donde los miembros desarrolladores trabajen para lograr un objetivo que debe ser percibido como común a todos. No se trata de "solidaridad", actitud humana superior, sino de sumar esfuerzos detrás de una meta que es buena para todos, "cooperación". Una meta cuyo alcance se logra con el aporte de todos los miembros, donde la calidad del producto depende de la calidad del trabajo de cada uno y donde el éxito del proyecto se convierte en un logro personal de cada miembro. Esta es la clave, funcionamiento del grupo de desarrollo.

Implementación:

1. Los responsables de los proyectos contratan y consultan a desarrolladores con experiencia para la construcción de un plan que contemple todos los aspectos de un proceso de desarrollo. La experiencia no se mide por la participación en numerosos proyectos; esta es una medida usada en general pero que ha mostrado muchas fallas. Hoy día es común recibir abultados curriculums, donde se describen largas trayectorias y estadías en numerosas empresas. Sin embargo, en un gran numero de casos los proyectos mencionados terminaron antes de dar a luz ningún producto. Por el contrario, una medida precisa es la participación en

unos pocos proyectos exitosos[1]. Estos desarrolladores serán miembros del futuro grupo y quienes conduzcan el proceso de desarrollo.

2. El líder del proyecto y los desarrolladores mencionados en el paso anterior seleccionan un proceso de desarrollo en forma consensuada.
3. El líder del proyecto y los desarrolladores acuerdan un medio de comunicación fluido y formalizado a través de la ayuda de artefactos gráficos (UML[%], PERT[(], etc.) y de herramientas que los implementan. Esta comunicación es en la dirección que marca el proceso de desarrollo.
4. Se definen los valores y las actitudes derivadas de ellos, sobre los que estará basado el funcionamiento del grupo. Se acuerdan las distintas formas de incentivación del trabajo en grupo.
5. Una vez conformado el grupo de trabajo, se comunican estos valores y las pautas que regirán la dinámica diaria.
6. Los responsables de la conducción del proyecto harán todo el tiempo trabajo de gestión, para que los desarrolladores puedan trabajar de la mejor forma posible. A cada momento se premiaran las buenas actitudes y se trabajara para erradicar las malas, mencionadas en la sección Problema. Los premios a las buenas actitudes deben ser simbólicos pero consecuentes con el espíritu de las reglas del grupo. Deben demostrar la gratitud de los conductores y el compromiso para con el miembro del grupo. Por ejemplo, una palabra de aliento y un "gracias" dicho en el momento justo. Los premios materiales como souveniers o invitaciones a compartir algún deporte o juego, utilizadas por algunos lideres de proyecto, son vistas por los desarrolladores como actitudes infantiles y carentes de compromiso por parte de quienes los conducen. En definitiva un pequeño show mediático. Tres son las cosas a no olvidar a la hora de mantener a un empleado motivado y dando en su trabajo lo mejor de sí: una buena remuneración, buenas relaciones humanas y tareas profesionales interesantes.

Variantes :

Existen publicaciones donde se propone atacar este problema con la colaboración de profesionales psicólogos y especialistas en relaciones humanas. Pueden ser de ayuda, sin embargo consideramos que la clave esta en la comprensión del problema por parte de los actores mencionados en esta presentación.

Consecuencias:

[1] Proyecto exitoso = realizado en tiempo y forma, con el presupuesto asignado.

[%] Unified Modelling Lenguage.

[(] Program Evaluation and Review Technique.

Ventajas
La diferencia entre un ambiente de desarrollo donde se considere de importancia el factor humano, y en particular estas actitudes; y uno donde no, es la misma que existe entre el éxito y el fracaso.

Desventajas
Ninguna.

Usos Conocidos:

En el año 2000, en la empresa TYSSA del grupo Telefónica de Argentina, el grupo de desarrollo de la Gerencia de Sistemas realizo una experiencia de este tipo con muy buenos resultados. Se desarrollo una "Intranet para Hoteles", trabajando en grupo cooperativo, ocupando roles y con una actitud positiva de todos los desarrolladores y la Gerencia de Sistemas. Todos los miembros habían compartido una formación de varios meses a través de cursos y seminarios internos. Todos ellos compartían la visión presentada en este trabajo.

Ver También:

Organization Patterns for Teams, de Neil B. Harrison publicado en [ref. 4], donde el autor ataca un aspecto de la formación de grupos de desarrollo. Estos patrones proponen soluciones similares a la aquí propuesta, sin analizar las fuerzas (actitudes) que Programmer's Attitude ataca.

EPISODES : A Pattern Language of Competitive Development, de Ward Cunningham, publicado en [ref. 4], trata aspectos vinculados con la organización de un ambiente de programación en relación con el mercado.

Patterns for designing in teams, fue presentado por Charles Weir en [ref. 6], en este trabajo se presentan cinco patterns para organizar el trabajo en grupo, asumiendo que "los miembros son cooperativos, inteligentes y constructivos en su relación laboral".

Reconocimiento

Guillermo Pantaleo agradece la cooperación, la muy buena disposición y la actitud profesional, a los miembros del grupo de trabajo de TYSSA, mencionado como caso exitoso en la sección Usos Conocidos.

Referencias

1. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "Pattern-Oriented Software Architecture - A System of Patterns". Wiley and Sons, 1996.

2. Alistair Cockburn , Humans and Technology , "d (Ad)/ d (Hf) : Growth of Human Factors in Application Development", http://alistair.cockburn.us/

3. Ivar Jacobson, Grady Booch, James Rumbaugh, "The Unified Software Development Process", Addison-Wesley, 1999.

4. J. O. Coplien, J. Vlissides, and N. Kerth, eds., Pattern Languages of Program Design, Vol 2. Reading, MA: Addison-Wesley, 1996.

5. Frederick P. Brooks, "The Mythical Man-Month: Essays on Software Engineering", Anniversary Edition (2nd Edition), Addison-Wesley, 1995.

6. Robert C. Martin, Dirk Riehle, Frank Buschmann, eds., Pattern Languages of Program Design, Vol 3. Reading, MA: Addison-Wesley, 1999.

7. Kent Beck, Extreme Programming Explained: Embrace Change. MA: Addison-Wesley.

8. Barry Boehm, J.R. Brown, H. Kaspar, M. Lipow, G. McLeod, and M. Merritt, "Characteristics of Software Quality", North Holland, 1978.

9. Barry Boehm, "Software Engineering Economics", Prentice Hall, 1981.

10. Barry Boehm, "Software Risk Management", IEEE Computer Society Press, 1989.

11. Tom Demarco, Timothy Lister, Timothy R. Lister, Peopleware : Productive Projects and Teams, 2nd Ed.Dorset House Publishing.

12. Tuckman B. 1965. "Developemental sequence of small groups" Psychollogical Bulletin. Number 63 p384-399

13. Kimball Fisher, October 1997. The Distributed Mind: Achieving High Performance Through the Collective Intelligence of Knowledge Work Teams

14. Teamworks, University of Illinois, Urbana-Champaign, http://www.change-management-toolbook.com/teamwork.htm.