NOTE: refers to a couple of other patterns ('see also').

NOTE Sebastian Spier (2021-12-11):
The content from this doc has been ported to our repo. The pattern is called Reluctance to accept contributions.

Title: Reluctance to accept contributions

Problem
The core team that owns a shared asset is reluctant to adopt contributions of contributing business divisions, because this implies adoption of the maintenance responsibility for the contributed software as well, a burden the core team does not want to adopt.

Context
An organizational unit (e.g. a business division or development team) is using an inner source product (a 'shared asset'), and finds it is missing a certain feature that they need and would like to integrate into the shared asset. The core team that owns the shared asset is responsible for its design and maintenance. The potentially contributing organizational unit has no champion on the core team.

comments
(done) business unit != owner
(done) protectionism clause of core team is defined now as a force, not in context

Forces
- The core team is responsible for maintenance, and therefore accepting contributions implies accepting the maintenance burden of new contributions. The core team is reluctant to take on this burden.
- The unit that uses the asset requires the feature to be implemented for their product, but it is not clear where or how the feature "fits"--and thus they have difficulty implementing it because they lack knowledge of the shared asset.
- Contributors that are not part of the core team do not have the same knowledge as the core team has about the shared asset.
- The core team is offered a contribution that doesn't fit with their vision of the shared asset--the contributor doesn't understand all the intricacies that the core team needs to consider.
- Contributions are not suitable for the core team to handle, either because of quality problems (e.g. lack of architectural compliance) or that the contribution is too large--the contributor isn't aware of how to contribute.

Solution
Define a clear contribution process for potential contributors.
- Implement a "support warranty" of limited time duration (e.g. 30 days) during which the contributor fixes any issues that the core team finds.

- Define clear contribution guidelines that define a set of expectations of contributions.
- Define a contribution workflow that describes submission, review, and acceptance steps, and which specifies how contributions can be tested.
- Provide training through tutorials, documentation, and one-on-one guidance.

Resulting Context
Potential contributors have clear guidance available and process to follow so they know what to consider when preparing contributions. The core team is better able to handle incoming contributions.

See Also
Support Warranty, Negotiate Contributions, Architecture for Participation, Feature Advocate

note: these patterns are currently under construction

Known Instances
Philips Healthcare, “Feature Advocate” described in: VK Gurbani et al. (2010) Managing a corporate open source software asset, Communications of the ACM vol. 53(2)

Status
11 Oct 2016 - First draft
13 Apr - Reviewed & Revised

Author
Klaas-Jan Stol

Acknowledgments
Georg Grütter
Bob Hanmer
Udo Preiss
Padma Sudarsan
Tim Yao
Nick Yeates

====

OLD VERSION
Title

Reluctance to accept contributions

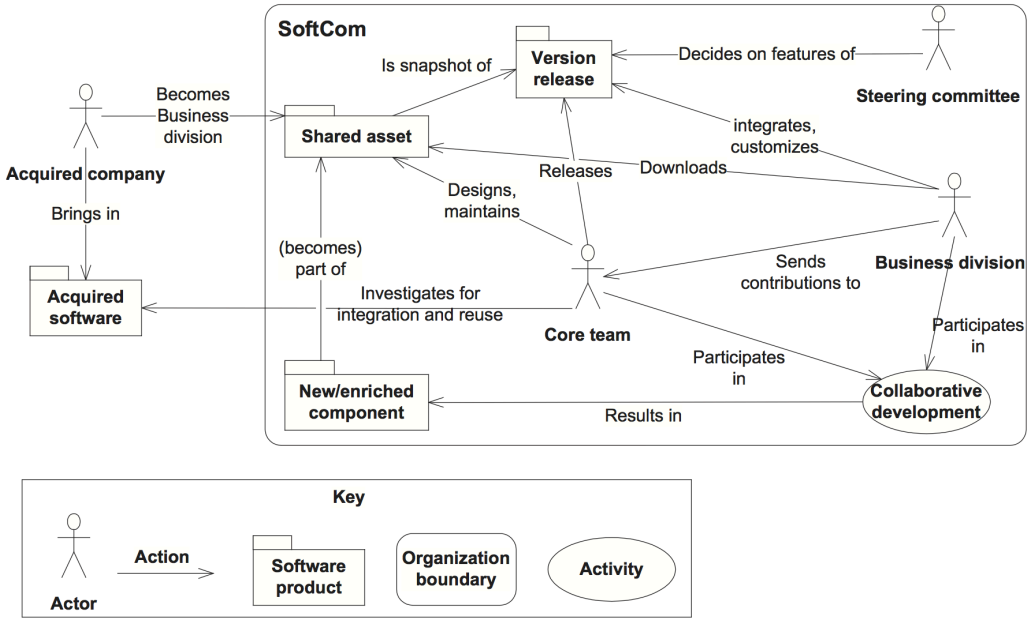
Problem
An InnerSource project's core team doesn't accept contributions because they tend to be of insufficient quality or do not comply with the shared asset's architectural principles.

Context
An organizational unit (e.g., a business unit or development team) is using an InnerSource project (also known as a "shared asset") and finds it is missing a certain feature they need, which they believe may also be useful to other users. The shared asset is owned by a core team who may be part of a different organizational unit. [think about maturity of product - does that affect the forces and solution]

Forces
- Contributors lack full perspective contributions; offered by users of the shared asset tend to fix only the users' specific use-cases, and don't consider other stakeholders.
- The core team doesn't want to adopt a maintenance burden for the contribution, as they are unsure about the quality of the code. [too many cooks spoil the broth?] [combine with:] The core team believes other teams' code is not as good, perhaps exhibiting the Not-Invented-Here syndrome. [is this mostly related to the fear aspect? fear of having to "pay" for others' mistakes]
- Contributions tend to suffer from quality problems. For example, a contribution may not comply with coding standards, architectural principles (e.g. design patterns), or makes assumptions that do not hold for other users (i.e., the contribution does not consider other user cases).

Sketch (optional)

this comes from Stol et al. 2011. Full ref here:
https://www.sciencedirect.com/science/article/abs/pii/S095058491100142X



Solution

- Write clear contribution guidelines that encourage appropriately-sized and well-documented contributions, and which define a set of expectations of contributions
- Co-development-contributors - pre-announce & negotiate a contribution
    - Contributors get help from a Feature Advocate from the maintainer/core team, who will help a contributor
    - Educate and coach maintainers to overcome their fears
    - Train and educate potential contributors on “best contribution practices”
    - To overcome the reluctance to accept the maintenance responsibility, the Thirty-Day Support pattern can be adopted.

Resulting Context

Potentially contributing business units have clear guidance available to them so they know what things to consider when preparing contributions [link to contribution guidelines solution aspect.] Contributions are discussed, negotiated, designed and implemented in collaboration with the core team ensuring that the contribution is based on real-world use cases (benefiting from the contributing business unit’s domain knowledge) and appropriately designed to fit the architecture of the shared asset. Contributions don’t arrive completely unexpected but are anticipated by the core team.

See also

Not invented here, (TBW) Thirty-Day Support, (TBW) Trusted Committer, (TBW) Fear of Contributions

Rationale (optional)

[why this solution works; go to play with friend; murder mystery; cluedo; resulting context; why was this the right solution; useful if solution is not obvious - can explain to reader why this solution works (perhaps in case of paradox)]

Known instances (optional)

Philips Healthcare, “Feature Advocate” described in: VK Gurbani et al. (2010) Managing a corporate open source software asset, Communications of the ACM vol. 53(2)

Status

11 Oct 2016 - Draft
9 Nov - Reviewed - first round

Author

Klaas-Jan Stol

Acknowledgments

Georg Grütter
Bob Hanmer
Padma Sudarsan
Tim Yao
Nick Yeates

Changelog

First draft: 11 October 2016
First review: 9 November 2016
First revision: 15 November 2016