

Reduce Software Project Failure Rates By Capturing Human Interactions

In the end, software applications are only there to support human work. Even a low-level, highly automated software application for (say) car numberplate recognition or payroll calculation is only there to meet the needs of the police officers or HR staff who ultimately set its initial parameters and use its output.

Yet most approaches to understanding and modelling human work have a major weakness – they offer reasonable support for capturing **H2S** interactions (between humans and systems), but are extremely weak when it comes to capturing **H2H** interactions (between humans and humans). Further, mainstream modelling techniques provide little of the context required to understand what truly goes in knowledge work.

When you consider the degree to which H2H interactions and human work generally are a fundamental feature of every workplace effort, does it not seem likely that this deficiency in requirements engineering is a major cause of the dismal failure rate of software projects?

On the success side, the average is only 16.2% for software projects that are completed on time and on-budget. In the larger companies, the news is even worse: only 9% of their projects come in on-time and on-budget. And, even when these projects are completed, many are no more than a mere shadow of their original specification requirements. Projects completed by the largest American companies have only approximately 42% of the originally-proposed features and functions. Smaller companies do much better. A total of 78.4% of their software projects will get deployed with at least 74.2% of their original features and functions.
<http://www.projectsart.co.uk/docs/chaos-report.pdf>

Consider, for example, the generalized software design framework known as the **Unified Modeling Language (UML)**. The UML permits human involvement to be notated via **Use Cases**, and expanded typically via **Sequence Diagrams**. Inside a Sequence Diagram there may be multiple **swimlanes**, each representing a stream of activity carried out by a person or software object. Interaction in the process is represented in the most primitive way possible – one can depict a message being sent from one swimlane to another.

And that's it. In both the UML and the other mainstream process notation, **Business Process Modeling Notation (BPMN)**, there is no obvious means to identify and record vital characteristics of the process that the diagram is supposed to depict:

- The personal characteristics of the people involved, for example the skills that qualify them to take part or involvement in related work;
- The organizational and administrative characteristics of the roles they are playing, for example their responsibilities and authority;
- The means by which they collaborate on an ongoing basis, for example to review or jointly author documents;
- Where and when they are intended or permitted to do essentially mental work, for example researching, evaluating, and analysing;

Reduce Software Project Failure Rates By Capturing Human Interactions

- How they are invited to become part of the process in the first place, and decide whether or not to do so;
- The manner in which their work is supported, reviewed, and controlled by management.

These human aspects of work are not tangential to a software design, but integral to it. Every requirements engineer knows that success of a system is closely linked to its support of the corresponding business processes. So key aspects of a business process such as those described above should be documented in the core diagrams on which the software will be based. Without this being done, it is unlikely that the system will succeed – and indeed very many software systems do fail, as reported decade after decade by researchers.

Human Interaction Management (HIM)

So understanding human involvement with a software system means *understanding collaborative business processes that include human interactions*. This is not as simple as it seems. It doesn't take much thought to see that business processes involving human interactions are not of a "mechanistic" kind – the kind of processes that your IT departments defines once and the organization then repeats ad infinitum with only limited variations each time.

Humans are usually there in order to bring uniquely human skills to bear – their knowledge, judgement and communication skills. And much of the time, they use these abilities to *work out what to do next*. In other words, they collaborate to define and re-define the business process as they go along, adjusting the participants, activities and interactions as necessary in order to get the job done. Hence modelling the business processes that involve human interactions needs a process design technique suited to work that not only is fundamentally collaborative but also that is carried out differently on each occasion.

Such process techniques and technologies do not set out to constrain work precisely. Rather, they:

- Provide standard process templates as a starting point;
- Ensure that general over-riding controls are in place;
- Allow the work to be integrated with management practices; and
- Automatically record and distribute audit trails for the work.

The process design does *not* pre-determine exactly the actions and interactions to be carried out in practice. Rather, the actions are determined by the individuals responsible for handling each case, on a case-by-case basis. In effect, the process participants collaborate to shape the evolution of each process, as part of the work itself.

The theory of **Human Interaction Management**, or simply **HIM** [see <http://human-interaction-management.info>] terms such collaborative, adaptive processes **human-driven**. In the free process notation and execution software for HIM [HumanEdi](#), human-driven processes are known as **Stories**. HIM provides a set of principles and techniques for the definition of Stories.

Reduce Software Project Failure Rates By Capturing Human Interactions

Case study: energy demand management

Here is an example Story, drawn up for a software vendor that provides solutions for energy demand management. In particular, their software both:

- Meters energy usage by individual facilities; and
- Tracks the current price of energy from various sources.

This enables organizations to proactively optimize their sources of energy supply on an ongoing basis. Since at times of peak demand some energy suppliers actually offer cash rebates to customers who reduce or suspend usage, energy supply can even be turned into a source of revenue by responding quickly to energy market changes.

Smart, but what was even smarter was the recognition by this software vendor that *using their products is effectively a collaborative business process* – so the best approach to implementing their products is to integrate them with support for human-driven processes.

What I will show here is not the Stories used in production, but an initial outline Story drawn up simply in order to illustrate the principles. Here is an overview diagram:

Reduce Software Project Failure Rates By Capturing Human Interactions

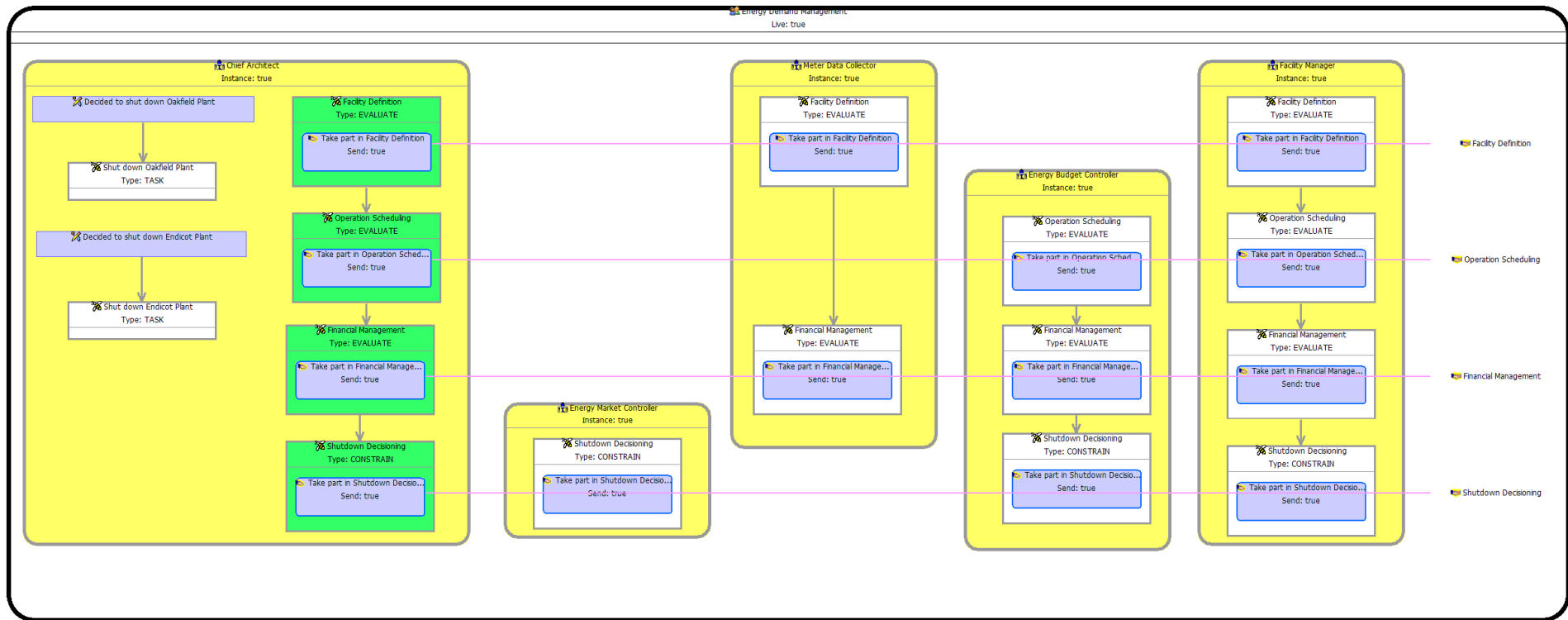


Figure 1: Overview of an outline Story for energy demand management

Reduce Software Project Failure Rates By Capturing Human Interactions

The diagram shows the **Roles** involved in the Story:

- Chief Architect
- Energy Market Controller
- Meter Data Collector
- Energy Budget Controller
- Facility Manager

together with the **Interactions** via which the players communicate:

1. **Facility Definition** - What facilities have we got? Who is responsible for production from each facility?
2. **Operation Scheduling** – What are the demand patterns? When can each facility possibly be shut off?
3. **Financial Management** - What are the costs of each facility? What is our energy budget?
4. **Shutdown Decisioning** - in which the Energy Market Controller can warn of rising costs, offering to pay consumers if they switch stuff off, and the Energy Budget Controller can make suggestions about switching source of supply

Note that each Interaction includes different Roles. However, the Chief Architect, as the driving force behind the work, participates in all of them.

This is simplified from a real-world Story, in which the work would be divided into a number of separate stages. Each stage is what the theory of Human Interaction Management terms a **collaborative transaction**:

- Started by an Interaction that defines the work to be carried out;
- Executed via different actions in each participating Role, to handle things like document creation/exchange as well as integration with server systems of various kinds; and
- Concluded by another Interaction that assesses status and agrees on deliverables.

The Story shown here is simplified so that most stages of the Story consist simply of a single Interaction. The exception is the final stage, in which the Chief Architect is able to initiate shutdown of energy demand from specific facilities, an action enabled by business rules within the Story.

However, note that a **Human Interaction Management System**, aka **HIMS**, makes it easy to add new activities on the fly. You can add activities to your own Role explicitly, or it will be done automatically when (for example) you open a new document as part of your work in the process. If you wish to add activities to other Roles, you can then share the new Story definition with some or all of the other process participants, and make a collaborative decision as to whether or not you wish to start using the new version.

Reduce Software Project Failure Rates By Capturing Human Interactions

The work done by process participants – whether to carry out activities or define new activities – is done *on the desktop*, with communications between Roles typically being via email. This makes it possible for the Story to proceed without everyone involved needing access to the same server applications, and without a single organization needing to own the entire process.

However, key activities in certain Roles are tightly integrated with a server-side, rules-driven system for Energy Demand Management (EDM). The EDM system itself integrates with both:

- The operational systems in each plant that monitor and control energy usage; and
- Third party applications providing business intelligence on the energy market.

The EDM system uses sophisticated business rules to supply the human Story participants with the information they need in order to make decisions, and helps them effect the decisions that they make.

Here are some screenshots illustrating human usage of the example Story.

Reduce Software Project Failure Rates By Capturing Human Interactions

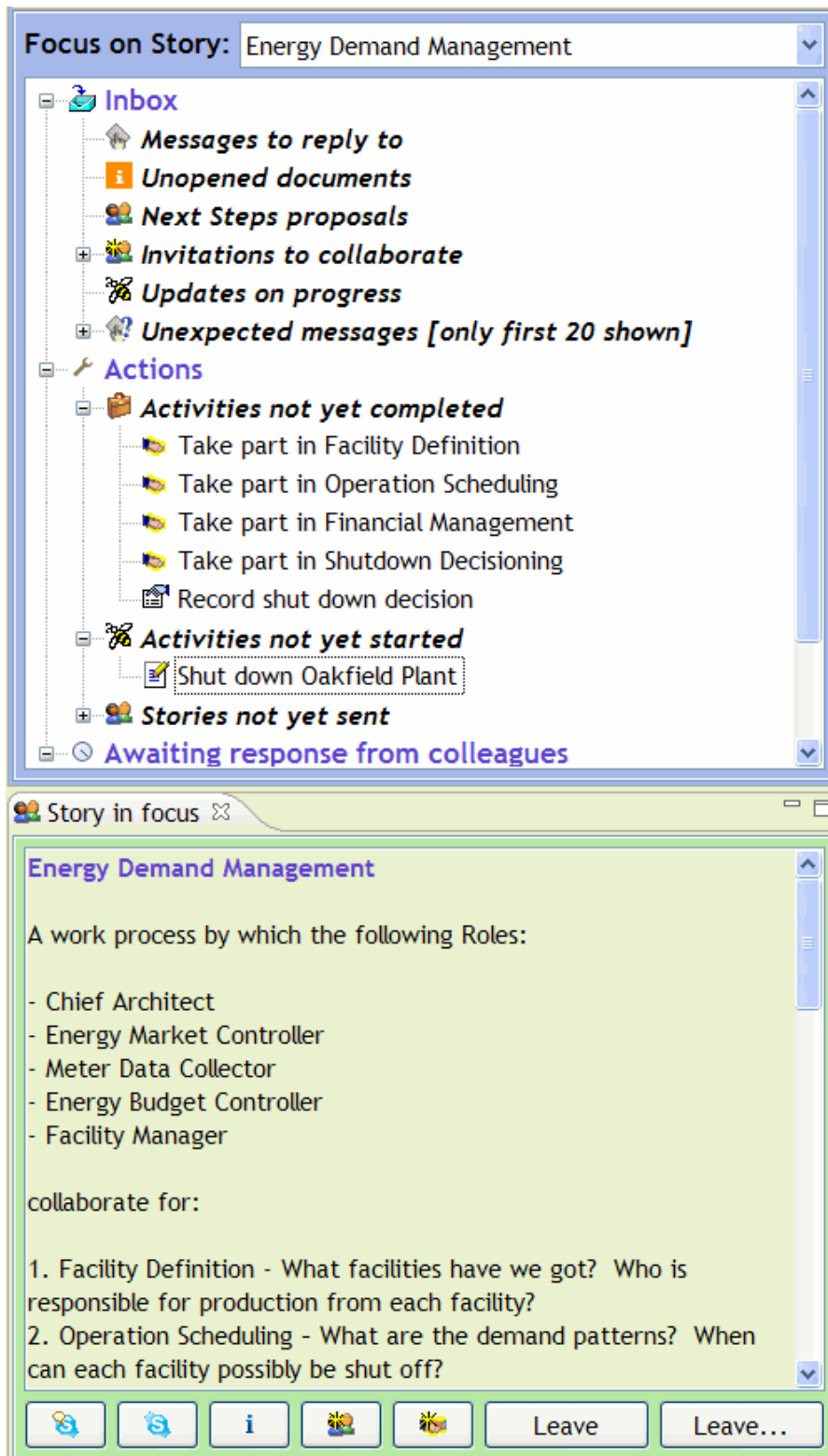


Figure 2: The Chief Architect's view of the example Story

Reduce Software Project Failure Rates By Capturing Human Interactions

Shown above is part of the desktop available to the Chief Architect, illustrating their options for participating in the process. In particular, they have various actions available. In this simplified Story, most actions are simply to participate in an Interaction.

Here are some screenshots showing the Chief Architect participating in Interactions:

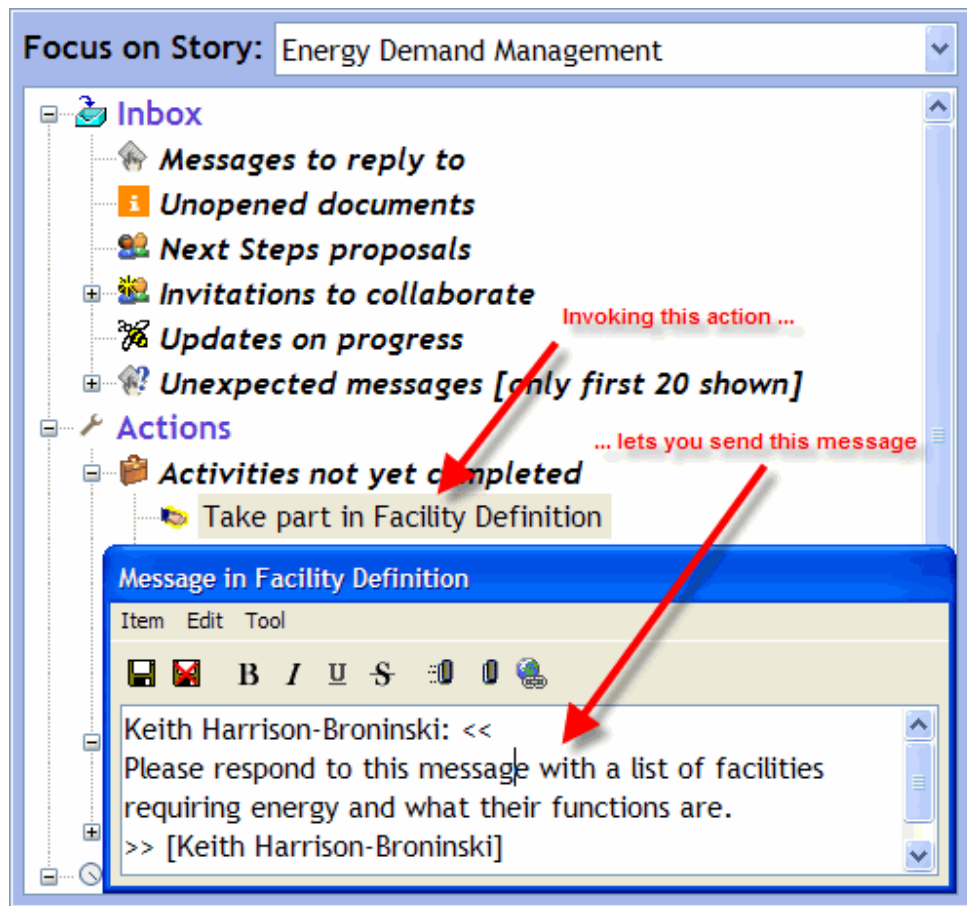


Figure 3: Chief Architect participating in the Facility Definition Interaction

Reduce Software Project Failure Rates By Capturing Human Interactions

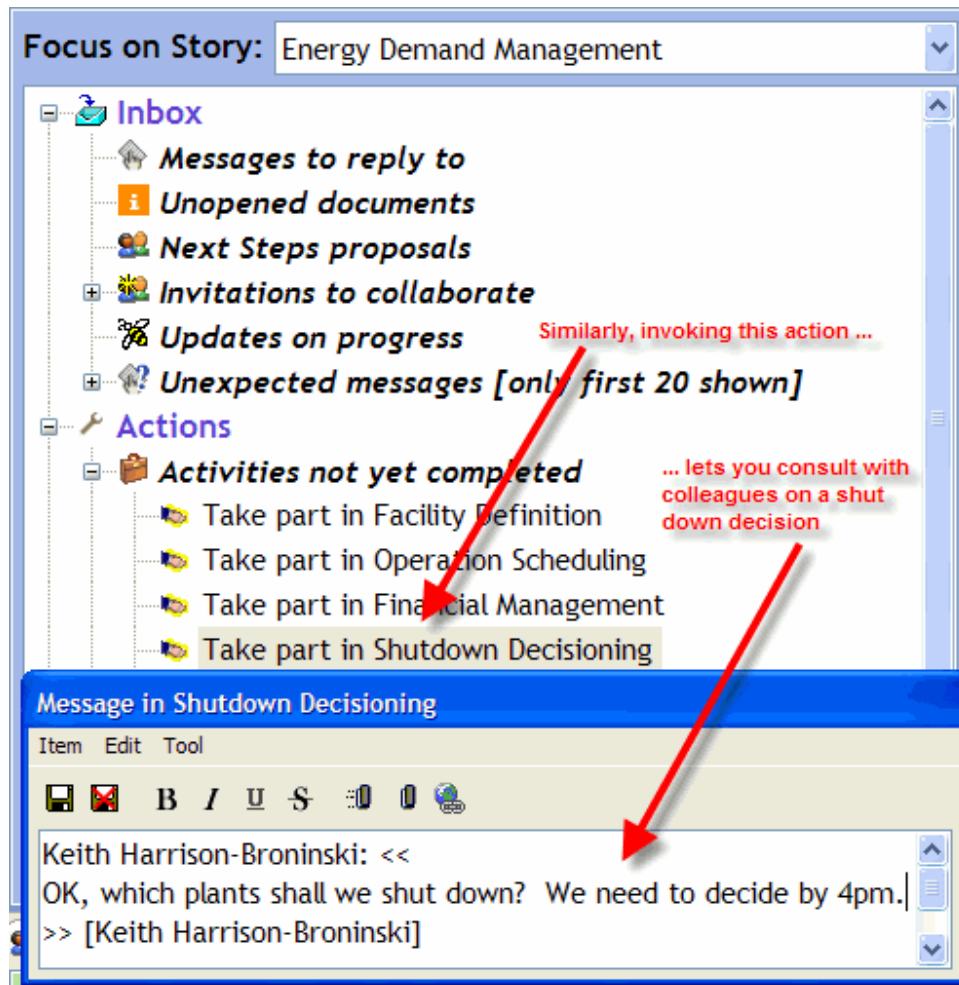


Figure 4: Chief Architect participating in the Shutdown Decisioning Interaction

The messages may be sent by the Human Interaction Management System using a variety of means. In general, email is used since this is the lingua franca in every workplace.

Further, use of email means that participants in the Story do not *need* to use the Human Interaction Management System to interact with their colleagues – they can reply to a message using a standard email program. Their response will form a natural part of the business process, with both message body and any attachments visible to all players involved in the Interaction. Similarly, Interaction conversations can be initiated using a standard email program.

There are some disadvantages to using a standard email program for process participation rather than using the process desktop. In particular, some of the fine version control over attached documents may be temporarily lost - and of course only messaging actions can be carried out via a standard email program (and these actions will not be controlled by business rules defined within the Story). However, participation from outside the process engine is a must for a modern working environment in which people:

- Interact with colleagues who work for different organizations; and

Reduce Software Project Failure Rates By Capturing Human Interactions

- Expect to be able to work using a handheld device in the airport lounge, without needing to load any special software onto this device.

Returning to the Story, as shown in **Figure 4: Chief Architect participating in the Shutdown Decisioning Interaction**, a key purpose is to enable collaborative decision making on energy demand management. Let's suppose that a decision has been arrived at by all concerned, to shut down one plant and not the other. The Chief Architect can then "make it so". The first step is to record the collaborative decision:

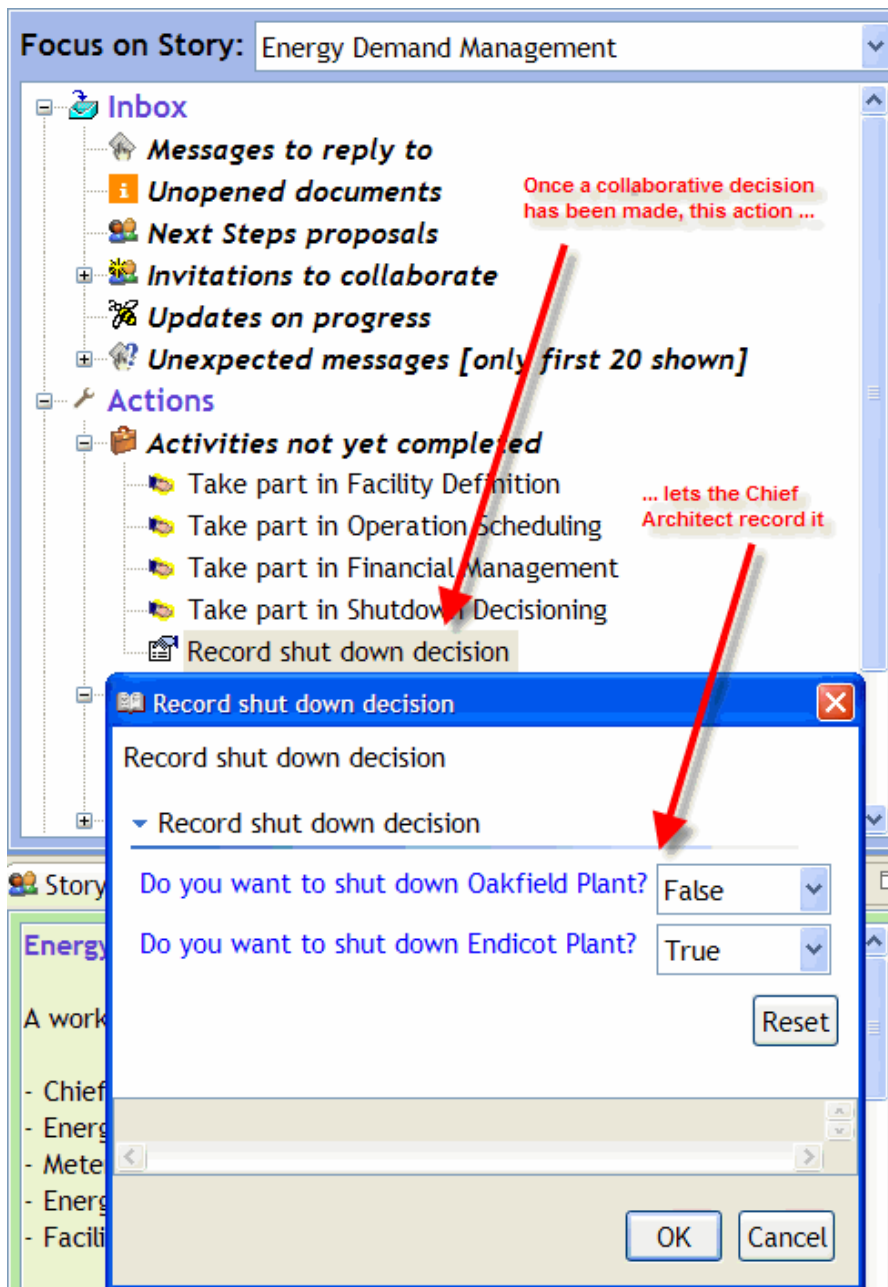


Figure 5: The Chief Architect records a collaborative shutdown decision

Reduce Software Project Failure Rates By Capturing Human Interactions

This triggers a business rule in the Story which makes available a particular action to the Chief Architect. This action integrates with the operational control systems to effect the shutdown:

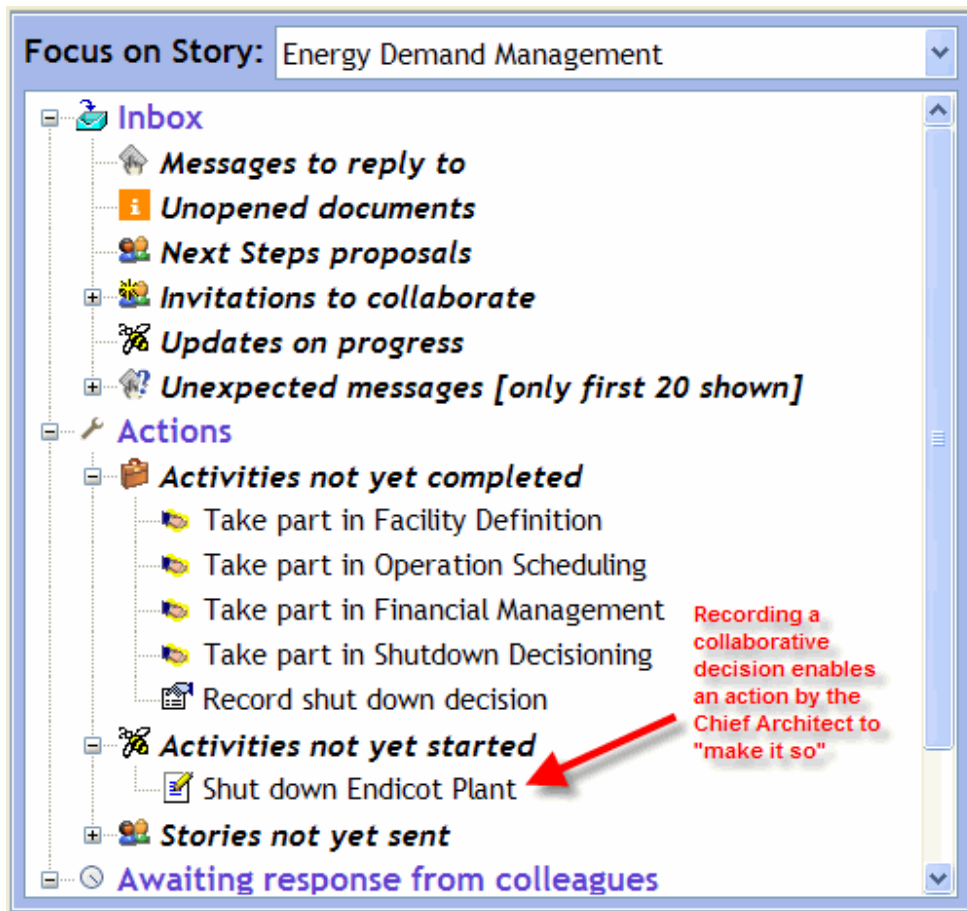


Figure 6: Chief Architect "makes it so"

Summarising the case study, the use of a Human Interaction Management System has implemented seamless integration between:

1. Energy market information sources
2. Operational facility metering and control systems
3. Business rule engines that integrate the above to recommend operational changes; and
4. The collaborative knowledge work (interaction work) required to ensure that energy usage is optimized without causing business disruption.

Smart energy management is made both smarter and safer by integrating it with process support for interaction work.

Reduce Software Project Failure Rates By Capturing Human Interactions

TAKE AWAY: Getting a handle on H2H

It is quite likely that many software project failures are directly attributable to requirements engineering failure – in particular, to the failure to capture the human-to-human interactions that are at the heart of much knowledge work. The problem is that analysts are trying to capture human work processes via techniques such as the UML that were only ever intended for designing software.

To improve the success of software projects, we need new techniques for business process design – techniques that properly support knowledge work and knowledge worker interactions.

For more information about **Human Interaction Management**, aka **HIM**, see <http://human-interaction-management.info>.

Author

Keith Harrison-Broninski is a consultant, writer, researcher, and software developer working at the forefront of the IT and business worlds. He is author of the landmark book "**Human Interactions: The Heart And Soul Of Business Process Management**" (Meghan-Kiffer Press, 2005, www.mkpress.com/hi), described by a BP Trends review as "*the overarching framework for 21st century business technology*," and by the BPM Group as "*a must read for Process Professionals and Systems Analysts alike*." Keith is also a contributing "thought leader" to the BPM Group book "In Search Of BPM Excellence" (Meghan-Kiffer Press, 2005, www.mkpress.com/bpmg.html).

Along with his research and consulting work, Keith is the CTO of Role Modellers Ltd, whose company mission is to develop understanding and support of collaborative human work processes across industry, a field that Keith has pioneered with his work on Human Interaction Management [see <http://human-interaction-management.info>]. Role Modellers' free software, [HumanEdj](#), is the reference implementation of a **Human Interaction Management System**.

Find out more about Keith and his work at <http://keith.harrison-broninski.info>.