A Backlog Model For Management of the Full Value Stream

Introducing the "Rock Crusher"

contributed by Steve Adolph

Contents

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements. No other part of the work so cripples the resulting system if done wrong."

No Silver Bullet: Essence and Accidents of Software Engineering. Computer
FP Brooks Jr - IEEE Computer Society, Washington, DC, 1987

Introducing the Rock Crusher: Making ALL Work Visible

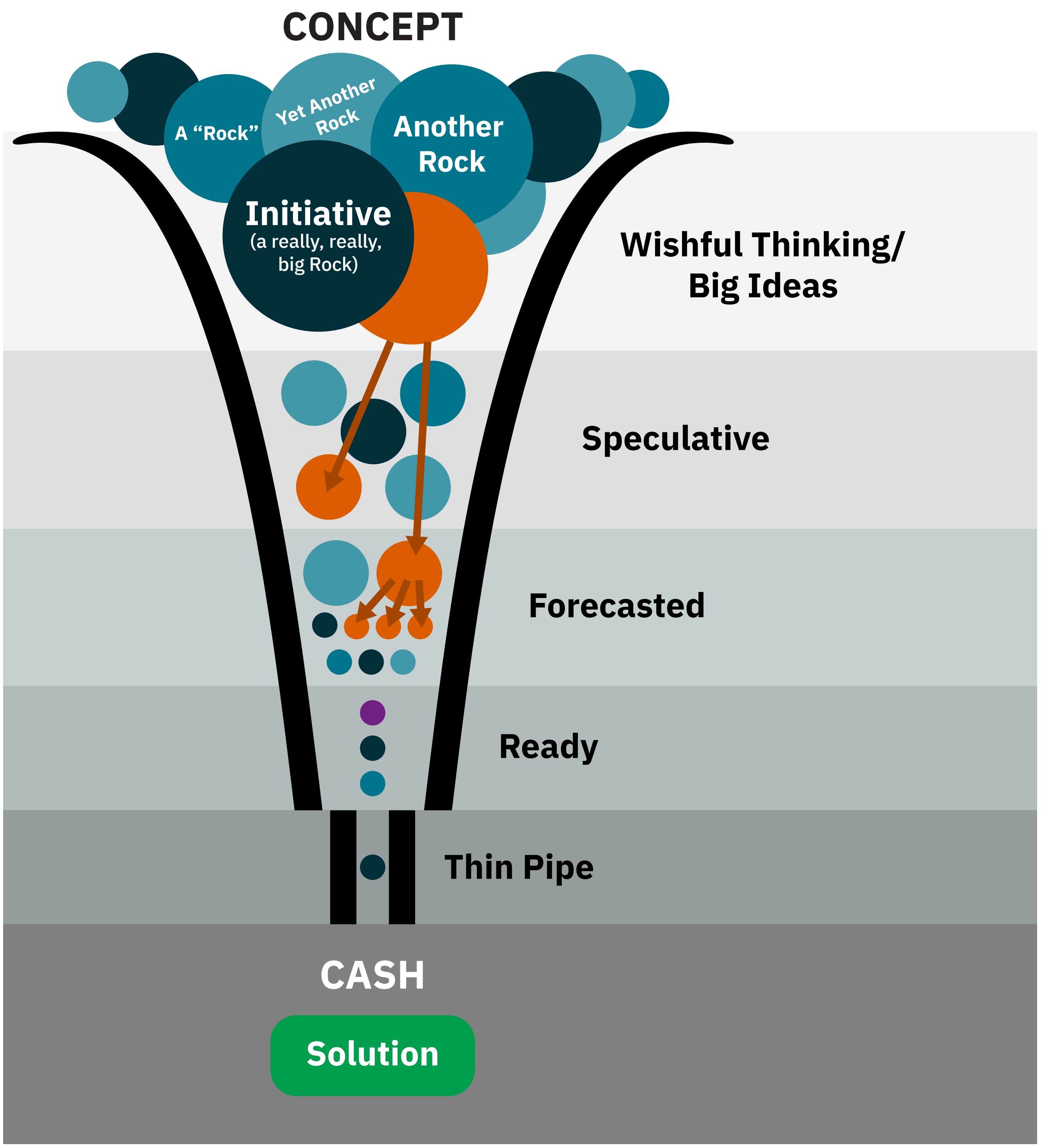
Nearly forty years ago one of the grand elders of the software industry, Frederick Brooks, called out the hardest part of building a software system as – "...deciding precisely what to build..." The customary mental model of the backlog and traditional Agile Methodologies often encourage us to make the work of "... deciding precisely what to build ..." invisible and unmanaged.

Most traditional Agile methodologies imagine the value stream as beginning and ending with a Product Owner. Work that does not directly contribute to the creation of working software is shunned. The work of "deciding precisely what to build" is kept hidden. The result is that, rather than the backlog enabling a flow of value, the backlog becomes a reservoir impeding the flow. To be a truly Agile organization, we need to visualize and manage ALL the work required to deliver a software system/solution, especially that hardest part of figuring out what to build.

To visualize and manage all the work go from "concept to cash," reimagine the shape of a backlog as a conical "Rock Crusher." Large conceptual "Rocks" are dropped in at the top, and the small pebbles and gravel that generate cash emerge in a flow from the bottom, delivering small increments of a solution. Larger abstract conceptual backlog items are progressively "crushed" – that is analyzed, designed, implemented, verified and validated – into smaller and more concrete work items.

The Rock Crusher is a metaphor for good backlog management practices that visualizes all the team's work. The Rock Crusher is an integration of common patterns that many teams are using to better visualize and manage all the work flowing through their value streams. Some organizations even refer to their processes as "Rock Crushers." Many of Rock Crusher concepts can be seen in second generation methodologies such as the Scaled Agile Framework and Disciplined Agile.

In the absence of such guidance, many organizations have reverted to Agile-waterfall hybrids. These hybrids may offer some technical improvement for development, but they do not improve the overall flow of value which is critical for the Agile Business.



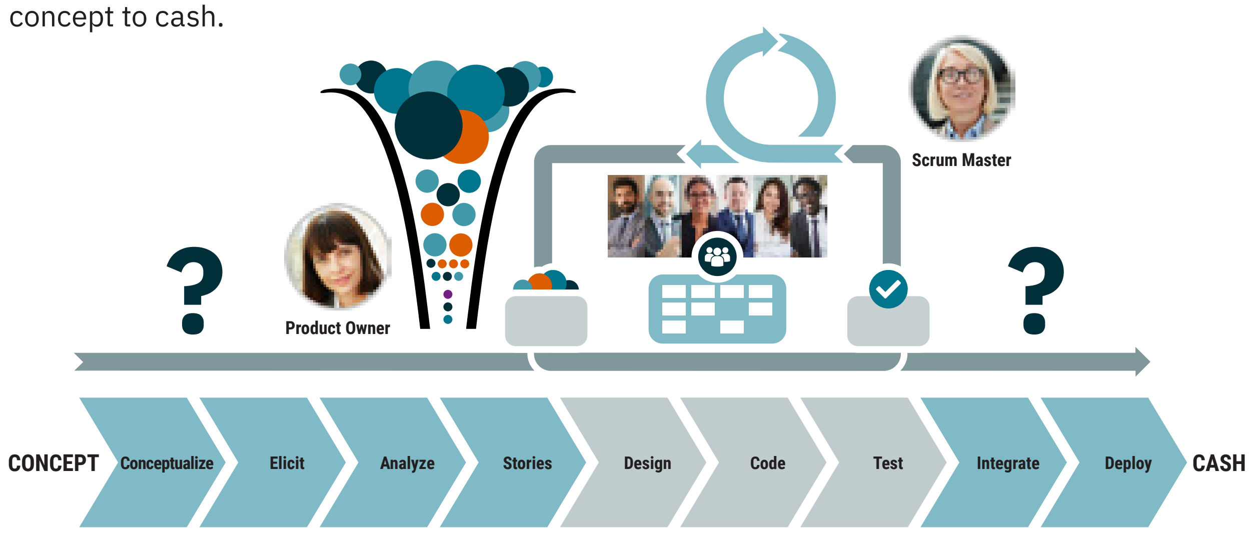
The Rock Crusher as Model for Backlog Management

The Rock Crusher is different from the classic "stacked plates backlog model" because the highest priority Ready work falls out at the "bottom" – the fine-grained results of the crushing process that lead to the Solution.

An Overlay on the Value Stream

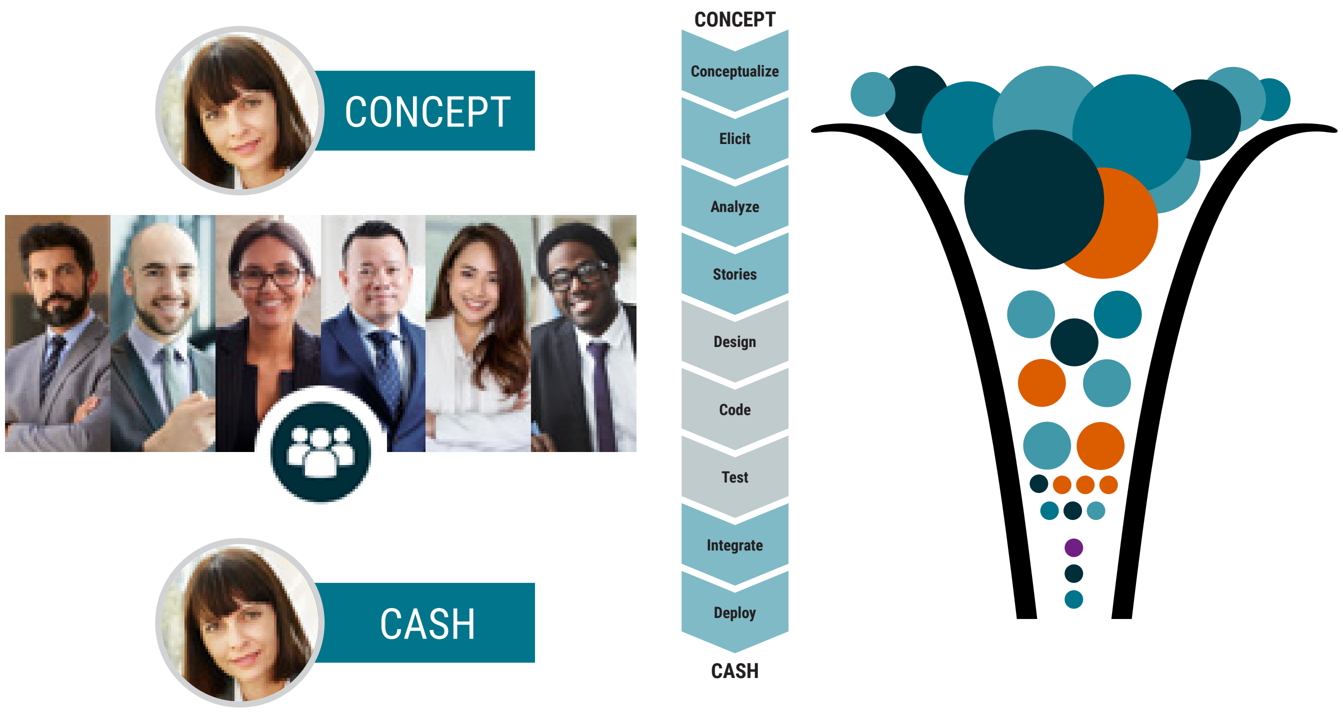
The Rock Crusher is a metaphor that visualizes backlog management as managing a flow of work through the entire value stream. Classical Scrum provides no guidance for what happens beyond the Product Owner and yet, for most development organizations, the value stream – what it takes to go from concept to cash – extends well beyond the Product Owner.

The Rock Crusher provides a model to visualize the entire value stream and all the work processes necessary to go from concept to cash.



Code-Centric View of the Backlog

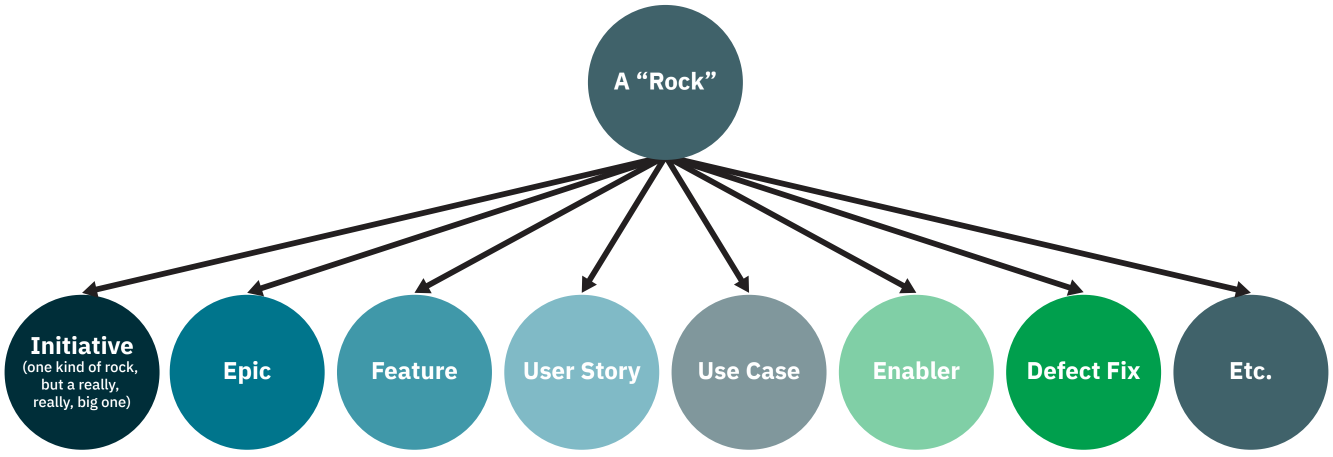
The Rock Crusher provides a model to visualize the entire value stream and all the work processes necessary to go from concept to cash.



The Rock Crusher Visualizes the Full Value Stream, Not Just Coding Activities

Rocks

The Rock Crusher crushes Rocks. A Rock is a synonym for a Backlog Item. A Rock can be any kind of work a team may have to perform, from a large initiative, to a user story, to a defect fix.



All Types of Work Are Represented as Backlog Items (called Rocks in this metaphor)

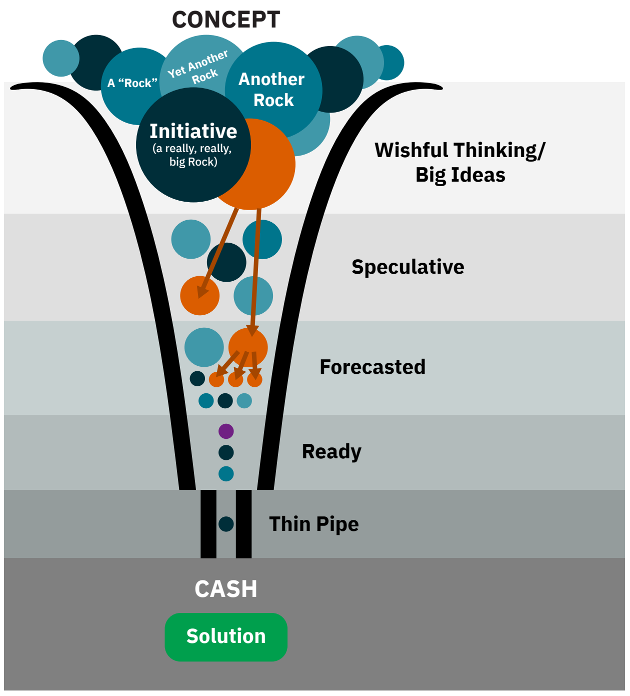
Crushing

Crushing refines and elaborates a larger abstract parent backlog item into smaller, more concrete, backlog items by adding verified knowledge.

Crushing and Splitting

The use of "crushing" is distinct from the classical term "splitting." Many people use the term "splitting" as a synonym for analysis. Analysis is decreasing ambiguity, and increasing understanding, of a problem. Crushing is a process of progressive elaboration, where the resulting "children" are more concrete, and of smaller scope, than their parent. Crushing adds learning and new knowledge to the understanding of the solution.

Splitting is strictly about right-sizing a big backlog item that does not fit into an iteration. Generally, when splitting, the resultant backlog items are at the same level of abstraction as the parent. Splitting is an implementation necessity to create backlog items that fit into an iteration or sprint. No new knowledge is added.



Crushing Refines Large Abstract Rocks into Smaller, More Concrete Rocks

Crush Meeting

Backlog Refinement Meeting

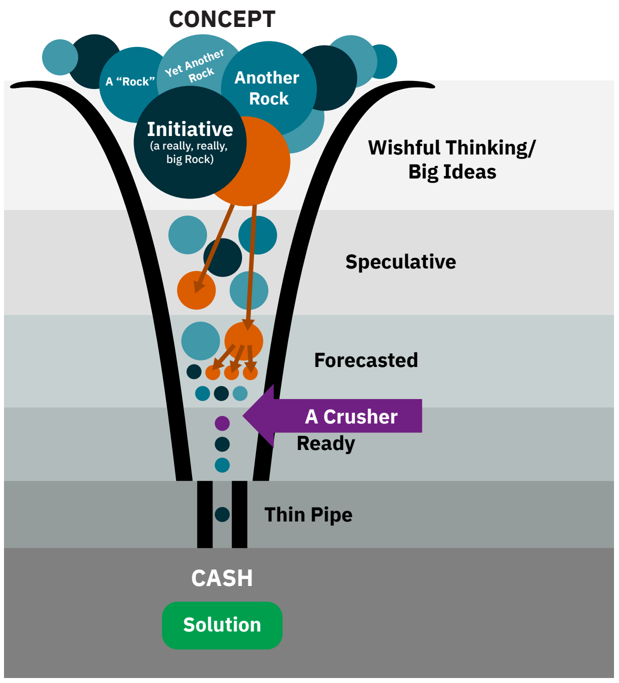
The Crush Meeting is a Rock Crusher synonym for the regular backlog refinement meetings a team holds. During a Crush Meeting, the team collaboratively crushes and refines Rocks, and advises the Backlog Owner on the economic value of Rocks. The team may be able to crush and refine many of the Rocks during the meeting, or the crushing and refining can be done as part of the Rock's final implementation. Other Rocks may require significant investigation, and the team will create and add "Crushers" to the backlog to capture and manage this work. During the Crush, what was learned from previous Crushers is evaluated and used to guide refinement decisions and prioritization.

Team Meets Regularly at Crush Meetings to Crush/Refine the Backlog

Crushers

A Crusher is a backlog item created to make learning "precisely what to build" visible during the crushing process. Crushers capture the important learning work that must be done in reductive transformation development processes when going from wild speculative concept to delivering a valuable working solution (cash). Typically, the output of a Crusher is a validated model which can then guide the crushing of a large Rock into smaller, more concrete items.

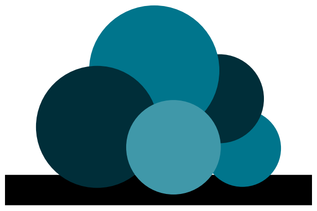
While Crushers do not necessarily result in "working software," they create the knowledge required to know precisely what to build. They create economic value by reducing the risk of building the wrong thing and aligning individuals on the ultimate solution. Crushers are objective and verifiable in that, like all well-formed backlog items, they include a means for testing and verifying that they are done – usually in the form of testable acceptance criteria. "Accepted" Crushers are presented at the iteration review. Crushers are not a justification for a stealth big up-front design (BUFD) process. They're intended to make the work to "... decide precisely what to build..." visible and managed, and not hidden.



Crushers Make Deciding Precisely What to Build Visible and Managed

Not All Rocks are Crushed – The Waste Gate

One feature of the Rock Crusher that is not shown in our sketch is the "Waste Gate." The Waste Gate is a metaphor for discarding Rocks that will not be crushed any further.



Waste Gate

Rock Crusher Waste Gate

Just because an item is in the backlog doesn't mean it will be implemented. At any time, all or part of a Rock may get discarded. The Rock Crusher has limited capacity and can only crush so many Rocks at one time. Rocks are prioritized or discarded based on the economic value they deliver. Discarding a Rock may occur early on when its realized an initiative will not create any worthwhile value, or later when a story does not contribute value. For example, a Rock which implements an edge case on a feature, may be discarded because the cost of implementing that Rock far exceeds the cost of the risk it is intended to mitigate.

The Waste Gate is usually implemented as part of a Crush Meeting (regular backlog refinement session). A Rock may end up passing through the Waste Gate – actively or passively.

Active	The team has made an explicit decision to remove the backlog item because it has become irrelevant or obsolete.
Passsive	The backlog item has "aged out." It has been in the backlog for so long that it is clear it has a low probability of ever being implemented. A good backlog management practice is to age backlog items.

It is up to the team on how they handle the backlog items ejected through the Waste Gate;

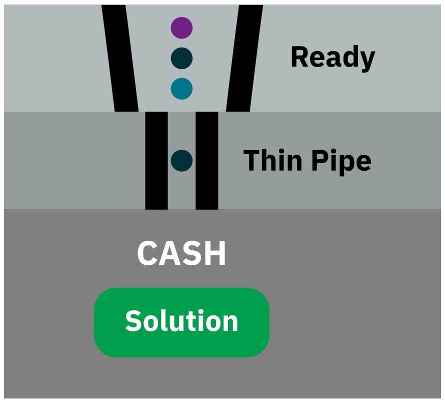
- They can simply delete the item and forget about it forever; or
- Use the concept of a fridge/freezer that serves as an archive for backlog items that aren't relevant now, but may come back in future.

The Waste Gate is an important Rock Crusher hygiene feature because leaving "obsolete" or forgotten Rocks in the Crusher simply plugs up the Crusher. Good backlog management hygiene demands constantly removing backlog items cluttering the backlog.

The Thin Pipe

The “Thin Pipe” is a metaphor for the subsequent steps in the product development process that pull Ready Rocks and deliver a Solution Increment. In a classic software development organization, the steps of the Thin Pipe may be:

1. Code;
2. Review;
3. Test;
4. Accept;
5. Deploy; and
6. Done.

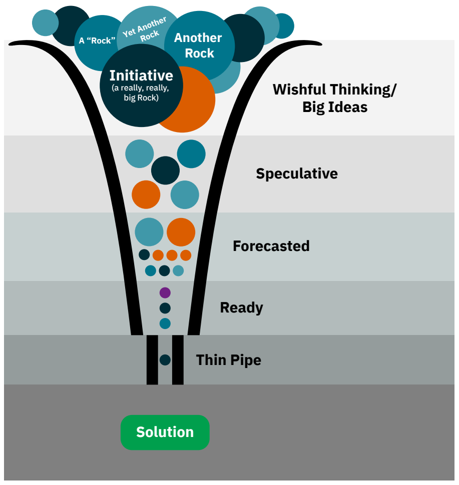


Thin Pipe Represents Subsequent Steps that Pulls Ready Rock to Complete Delivery of a Solution Increment

The Rock Crusher captures how Rocks can be transformed from Wishful Thinking to being Ready. Getting Rocks “Ready” means having well defined, valuable, serialized and prioritized work that is ready for the next stages of the product development lifecycle. However, once Rocks are Ready, it defers to the “Thin Pipe” for subsequent transformational and crushing stages.

The Rock Crusher Leads to Solution Increments

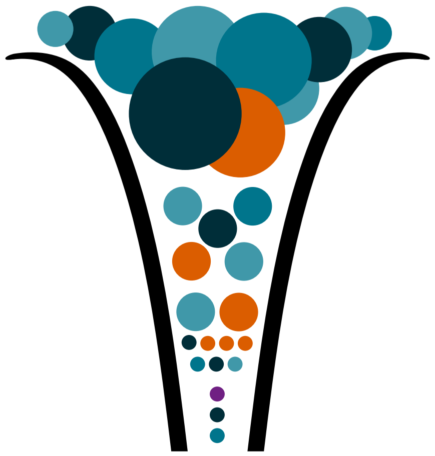
The ultimate economic goal of the Rock Crusher is to deliver a flow of the highest value solution increments. The solution is ultimately what is being built, what is desired by Customers, and what Stakeholders have an interest in.

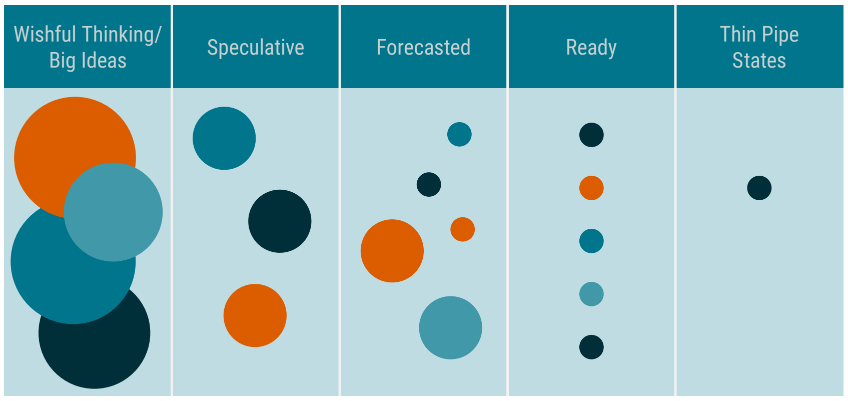


The Rock Crusher Leads to a Solution Increment

The Rock Crusher can be Implemented Using a Kanban

Most Agile lifecycle management tools use the stacked plates model to represent the backlog. However, the Rock Crusher can be implemented using a Kanban. Kanban is an ideal tool for implementing the Rock Crusher. Kanban visualizes work and through the effective use of Work-in-Progress (WIP) enables managing throughput.

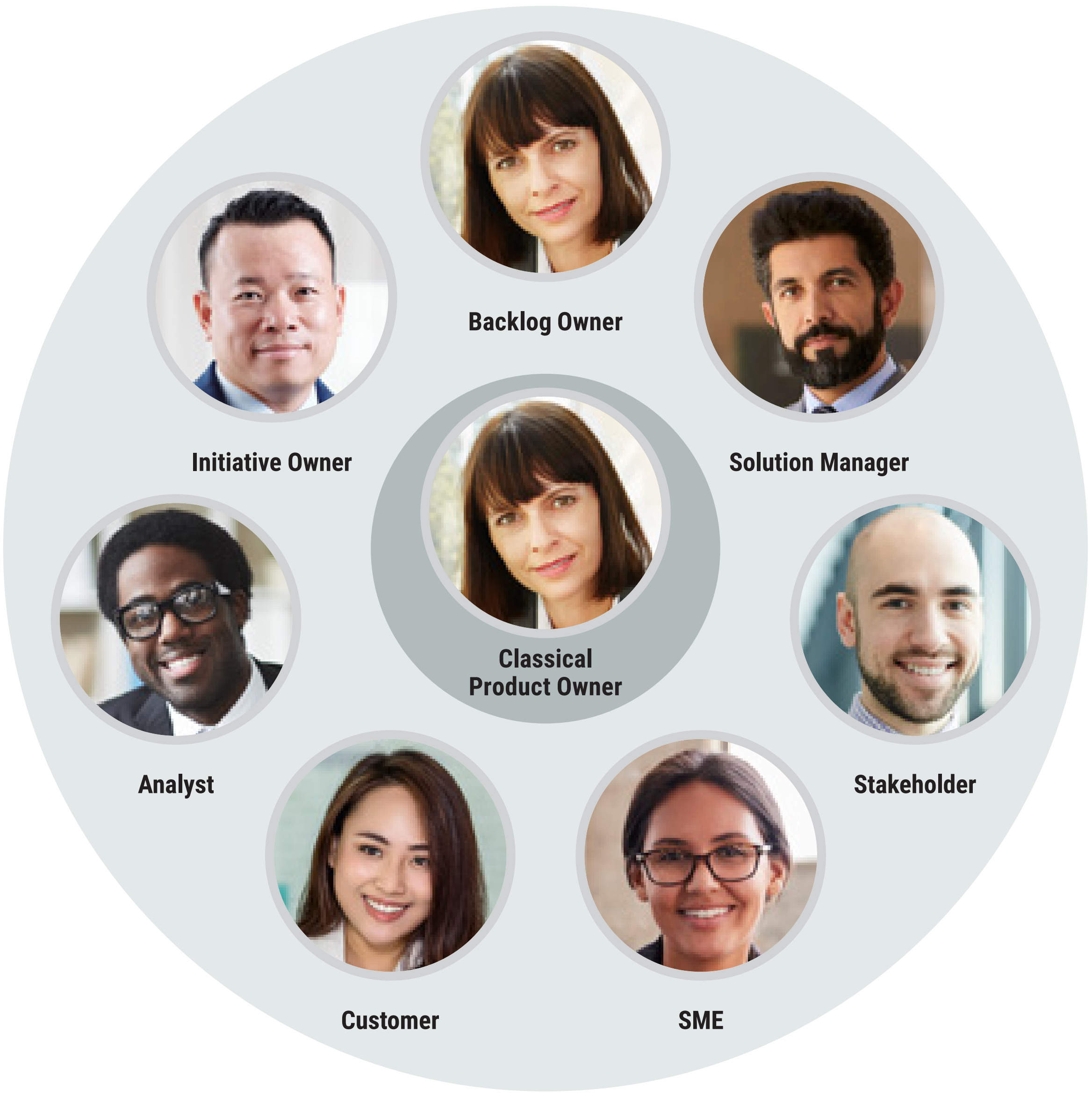




Rock Crusher Roles – It Takes a Village

Classic backlog management is usually performed as a collaboration between an omniscient and omnipotent Product Owner, and their development team. The reality is that for most development organizations this model does not work because the role of Product Owner is so demanding it cannot performed by an individual. The Rock Crusher “explodes” the role of the classical Product Owner into seven roles.

These roles, and how many individuals are required to perform them, greatly depends on the size and context of the development environment. For example, in a small, focused skunkworks-like environment, it may be possible for a single individual to play all seven roles as the classical product owner. In others, the roles may be split between two to three people such as a Solution Manager, a Backlog Owner and an Analyst.



	Customer	• The receiver or beneficiary of the solution. • May need to be informed about decision as to precisely what to build.
	Stakeholder	• Any individual who, at a minimum, must be consulted about precisely what is being built, and may also have decision making authority in deciding "precisely what to build." • A Customer may also be a Stakeholder if they need to be consulted about decisions on precisely what to build.
	Initiative Owner	• Accountable for getting an Initiative done. • Many different teams and individuals may have to collaborate to get the Initiative done, and, therefore, it is easy for an Initiative to "get lost in the process."
	Subject Matter Expert (SME)	• Someone who has deep knowledge of the relevant problem domain and/or relevant technology and development practices. • SMEs use their knowledge to advise other roles "deciding precisely what to build." • They are responsible for providing the expertise other roles may need to perform their jobs.
	Analyst	• Responsible for transforming the sometimes competing wants, hopes, interests and aspirations of the Customer, Stakeholder, Initiative Owner, Product Manager, and the contributions of the SME, into a shared, clear understanding of precisely what to build. • Responsible for progressively creating a better understanding of precisely what to build – the solution.
	Solution Manager	• A customer facing role accountable for which features make into a solution but doesn't have final say on prioritization of work for an individual team. • A Solution Manager is different from an Initiative Owner because the Solution Manager role is linked to the long-term evolution of the solution.
	Backlog Owner	• Has the final say on the sequencing /prioritizing of work for a team. • The Backlog Owner has the final say because they are accountable for ensuring the team is always working on the most valuable work.

Rock Crusher Horizons

Some suggestive "time horizons" – Ready, Forecasted, Speculative and Big Ideas have been added to the Rock Crusher model to suggest what the scope, granularity and time criticality of the backlog items may be.

What these horizons mean depends on the context of the Rock Crusher in your organization. For example, the time horizon for Forecasted may be the next sprint for a small product development team, while it may mean the next quarter for a very large program.

Example of a Rock Crusher time horizons for a small product development team:

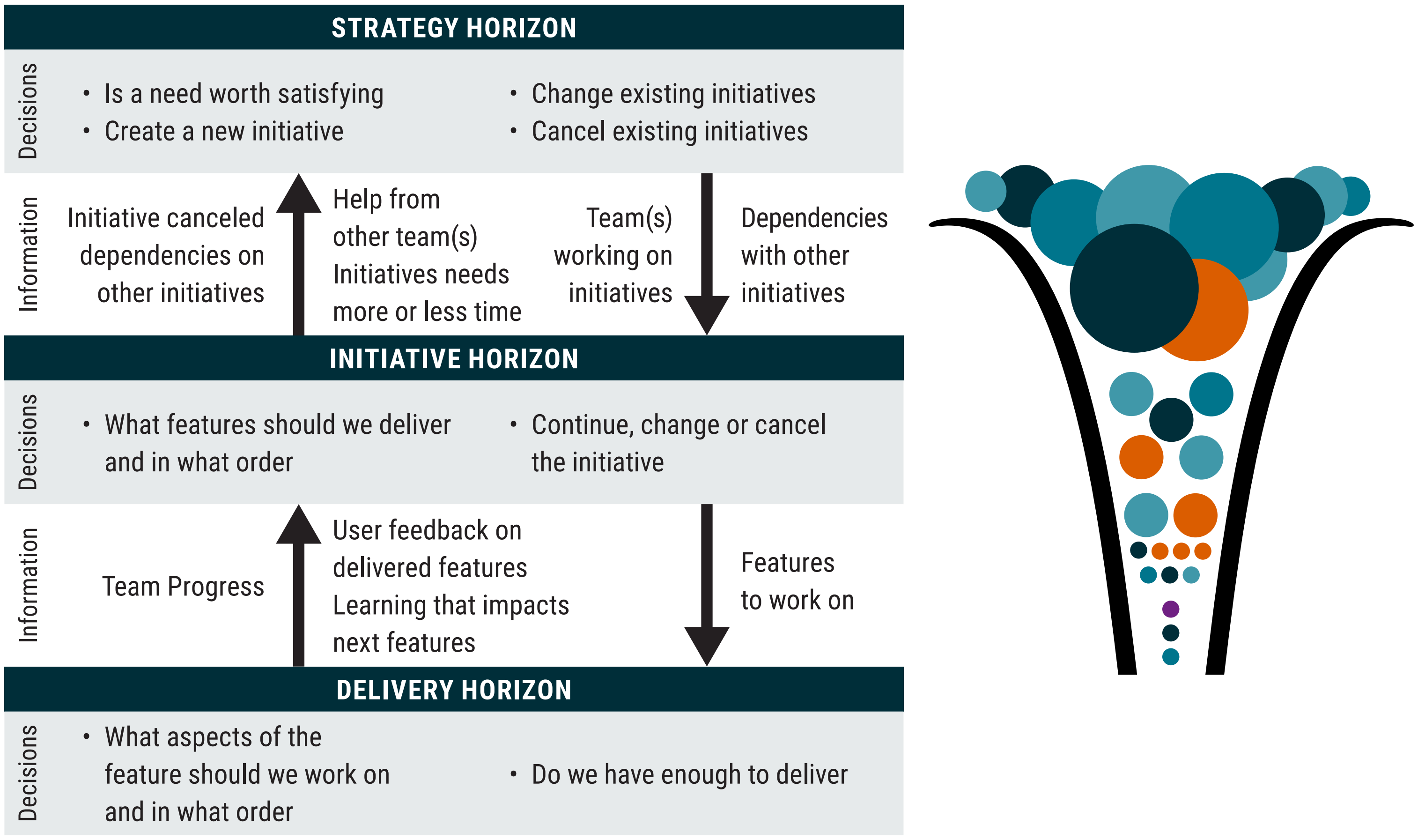
Ready	• Work that is "ready" for a team (e.g., a well-formed user story). • Small in scope, may only take a couple of days to do. • Work item is sufficiently well formed that a team can commit to delivering it in a single iteration or sprint. • In a Kanban world, they can satisfy their service level agreement (SLA) for work of this type and size. • Backlog items are forced rank prioritized.
Forecasted	• Work that is near ready with a high degree of certainty the team will commit to the work (e.g., a user story or a small size epic user story). • Upcoming on a roadmap, maybe a couple of weeks to a couple of months out in the future. • Work item is sufficiently well formed that a team can reasonably size it and know if they need to split it into right sized backlog items (can be implemented in a single iteration or sprint). • Backlog items are "near forced rank" prioritized.
Speculative	• Work that is not quite ready, with only a moderate degree of certainty that all or part of the work described by the backlog item is needed or will get done (e.g., moderate to large Epic User Story). • Broader and more abstract than Forecasted work. • Work items may represent a few weeks of work for a team and are at least a few months in the future. • Team can likely size story using t-shirt sizing (e.g., xs, s, m, l, xl).
Wishful Thinking / Big Ideas	• These could represent anything far off in the future, e.g., in two or three quarters or more. • The scope could be very large. • It isn't certain if the idea will pass vetting. • It's unlikely the team can size the item beyond a coarse t-shirt sizing (e.g., s, m, l). • These ideas are "pitchable," but not yet economically defensible.

At the “top” of the Rock Crusher, far in the future (“wishful” thinking) are “big chunks” that represent big vague ideas. They:

- Can’t be “force ranked” because there isn’t enough knowledge to do so.
- Haven’t been vetted.
- May represent early strategic ideas.
- Could be initiative-sized Rocks (in a large-scale environment) that are large in scope and fuzzily defined – the 30,000-metre view.
- May be loosely clustered into priority groups using the WSJF or MoSCoW criteria.
- May be discarded – quickly tossed through the “Waste Gate.”

The remaining chunks are then progressively “crushed” and elaborated into smaller, more concrete, chunks.

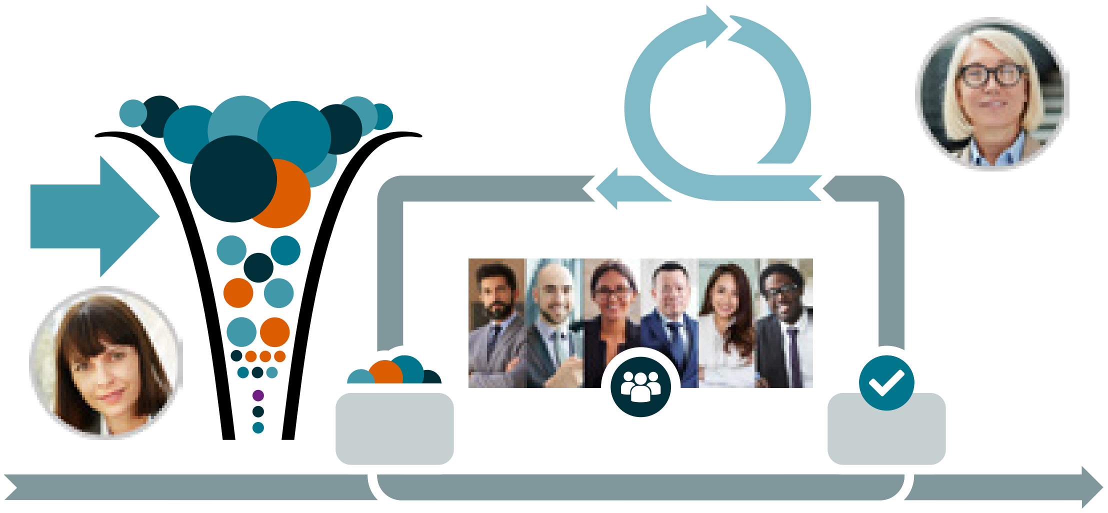
The Rock Crusher aligns well with rolling wave planning models such as the IIBA Three Planning Horizon Model. These planning models use learning cycles from lower levels to influence the decisions made at higher levels. Learning cycles at lower levels are faster (e.g., a two-week iteration) than those of higher levels (e.g., a 10-week SAFe PI at the initiative horizon). At the lower levels, the Rocks are small; up the horizon hierarchy, the Rocks become larger and more abstract.


STRATEGY HORIZON
Decisions
• Is a need worth satisfying
• Create a new initiative
• Change existing initiatives
• Cancel existing initiatives
Information
Initiative canceled dependencies on other initiatives
Help from other team(s) Initiatives needs more or less time
Team(s) working on initiatives
Dependencies with other initiatives
INITIATIVE HORIZON
Decisions
• What features should we deliver and in what order
• Continue, change or cancel the initiative
Information
Team Progress
User feedback on delivered features Learning that impacts next features
Features to work on
DELIVERY HORIZON
Decisions
• What aspects of the feature should we work on and in what order
• Do we have enough to deliver

Rock Crusher Modeling the IIBA Three Planning Horizon Model

The Rock Crusher is at the “Centre” of the Team

Backlogs are intended to facilitate the delivery of a flow of value. Unfortunately, many backlogs devolve into reservoirs rather than facilitating flow. Part of the reason for this is that most models of Agile methodologies portray the product backlog outside of the team. It is the Product Owner’s responsibility to maintain a ready supply of backlog items for the team. Therefore, the team doesn’t care how the Product Owner got the concept, performed the analysis, or how the design was created, just as long as they are not starved for stories. This view reduces collaboration, if not eliminating it all together.



Traditional View of the Backlog Outside of the Team

This view encourages a coding-centric view of Agility and is misleading because it implies the purpose of the entire organizational ecosystem is to supply a software team with near Ready backlog items which the team codes and tests into “working software.”

Therefore, rather than representing the backlog as a means for just “feeding” a development team, ALL the work required to create a product or service is called out and shows that the team is responsible for “running” the Rock Crusher – the entire value stream.

The Rock Crusher becomes a way of managing how value is created from ideas. The team is responsible for all the work which must be done to deliver an increment; not just code, test and deploy, but also ideation, analysis, design, and whatever else is necessary to “go from concept to cash.” Far too many organizations manage analysis and design work using a traditional stage gate model, or worse, where work is “off balance sheet” and “invisible.”

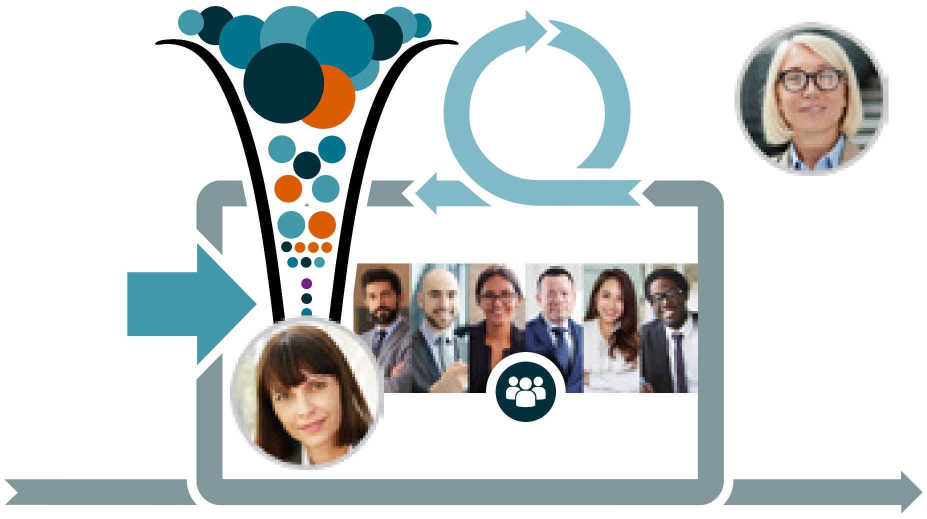
Placing the Rock Crusher at the centre of the team creates a visual where the whole team is responsible for ideation and delivery. The whole team not only builds the system but is also involved in "...deciding precisely what to build." The whole team collaborates on performing:

- Analysis;
- Design;
- Implementation; and
- Deployment.

A true cross-functional delivery team, not just a coding team.

Placing the Rock Crusher at the centre of the team brings team members who were in the closet, out. It encourages application of Agile thinking to analysis and design and whatever other processes are required to transform an idea into a valuable product or service. Placing the backlog at the centre of the team is a visual reminder of the collaboration necessary to deliver a system.

The Rock Crusher model works for teams big or small, and a team is all the individuals who work together to deliver a system or solution. While the Rock Crusher does define specific roles (e.g., Product Manager, Product Owner, Analyst, Initiative Owner and Subject Matter Expert), the Rock Crusher does not specify how teams should be organized. The Rock Crusher only defines collaboration between the roles.


Backlog at the Centre of the Team

Summary

The Rock Crusher is a metaphoric model for the backlog that tries to mitigate the deficiencies of the traditional “stacked plates” model. The Rock Crusher better captures and visualizes the ambiguity that should be inherent in a backlog.

The metaphor of “crushing” expresses how the Rock Crusher visualizes the analysis process.

Crushing is distinct from “splitting.” Crushing expresses the analysis process of progress elaboration. The results of crushing are work items (Rocks) that are smaller, more detailed, more concrete than their parent. Collectively, all the children of a “crushed Rock” should be a more detailed expression of their parent.

Like a traditional backlog, there is no commitment to a work item just because an item is in the Rock Crusher. There is no commitment to getting it fully “done.” Rocks do not have to be fully crushed, or even crushed at all, and there is a “Waste Gate ” on the Rock Crusher to eject Rocks that will never be implemented.

Learn More

Rock Crusher for Backlog Management

Read:

1. Introduction
2. Ceremonies
3. Implementing
4. Roles
5. Crushes/Backlog Items

Considered Harmful
Anti-Patterns
Re-acquainting
Work not Road Mapped

View:

Rock Crusher Infograph

User Story Infograph