Humans in the Black Box: A New Paradigm for Evaluating the Design of Creative Systems

Brad Spendlove and Dan Ventura
Computer Science Department
Brigham Young University
Provo, UT 84602 USA
brad.spendlove@byu.edu, ventura@cs.byu.edu

Abstract

Modular creative systems employ specialized modules that take on challenging responsibilities. Such modules can be expensive to develop and are often imperfect, leading to uncertainty in whether a system's poor results are due to module imperfection or larger systemic issues. Computational creativity research would benefit from a method for validating the design of creative systems before expending development resources implementing them. We present *ideal-module prototyping* (IMP), a design validation paradigm that serves these needs by using modules that delegate responsibilities to humans as an ideal against which to compare system expectations and performance. We relate an experiment we conducted in applying IMP to an existing creative system to demonstrate the insights that can be gained from this method. We argue for the widespread adoption of modular creative system design validated by ideal-module prototyping.

Introduction

Computational creativity research has as one of its goals the invention of computational systems that can be said to exhibit creativity. This is a lofty and challenging goal that approaches the limits of humans' understanding of our own mind and consciousness.

Given such a challenging goal, it is no wonder that successful computational systems that advance closer to that goal are complicated and at times difficult to understand themselves. Explainability is a hot-button topic in many stripes of AI research (Ehsan and Riedl 2020), as advanced machine learning algorithms increasingly exceed their own creators' abilities to predict and control.

Computational creativity researchers, who create systems that by necessity are complex and unpredictable, often face similar challenges. The complexity of computationally creative systems, coupled with the vast spaces of possible outputs in a given creative medium, can result in systems that produce low-quality output but whose internal design flaws are difficult to diagnose.

One approach to solving these issues is to implement a prototyping method that will reveal system flaws before they result in low-quality output. Another is to develop tools for more nuanced diagnosis of existing systems with low-quality output. The development of a useful paradigm for prototyping the designs of new creative systems and diagnosing the flaws of existing systems would be beneficial to the computational creativity community.

Modular Creative Systems

Modular creative system designs use specialized modules that compute tasks—such as knowledge lookup, transformation, combination, and evaluation—connected by higher-level logic that organizes the flow of information between them. The tasks implemented by these modules often represent challenging operations that attempt to reach human-level performance. For example, a module could attempt to represent knowledge similarly to a human or evaluate a creative artifact as a human would. Even tasks like computer vision that do not seek to produce the same output as a human still often need to function on a level of performance equal to that of a human.

Because these tasks are extremely challenging, the modules that implement them commonly represent the weak links in a creative computational system. Conversely, if such a module is working as intended, but the system as a whole still produces low-quality results, it is important to make that distinction and not expend effort on modifying a module that already functions well.

With a number of interoperating parts, many of which are attempting to complete such challenging tasks, it can be difficult to diagnose problems with a modular creative system as a whole. When the quality of the system's output is low, what is the cause? The ability to tease apart the interplay between modules and higher-level control would be useful both to reduce diagnostic complexity and to analyze tricky modules without conflating issues from other modules or information flow.

Ideal Modules

One approach to evaluating a modular creative system is to compare its performance to a version of that system with ideal modules. Each module that is replaced with an ideal version is one that cannot be the source of flaws in the system's output. If all of the system's modules are ideal, then any flaws in its output must be the result of how those modules are interconnected or some other holistic aspect of the system design. Alternatively, if the system yields high-quality output with ideal modules, then any reduction in out-

Proceedings of the 11th International Conference on Computational Creativity (ICCC'20)
ISBN: 978-989-54160-2-8

put quality when the same system uses imperfect modules can be attributed to the difference in module quality.

For some tasks such as mathematical operations, existing computational modules represent the ideal. However, many tasks that are useful in computational creativity do not currently have ideal computational implementations. We propose that for these types of modules, a useful definition of an ideal module is one that delegates completion of the task to a human. This conception of ideality serves creative systems whose modules or output are intended to mirror human performance, be compatible with a human audience, or achieve human-level quality.

We will refer to such modules as *human-delegated modules*, to indicate that they complete tasks by presenting the task to a human to solve. The inclusion of human-delegated modules in a creative system would obviously discount it from being considered an autonomous computationally creative system. However, we propose that testing a creative system using ideal modules, even if they are not computational, is a powerful tool for prototyping and evaluating the algorithmic validity of the system's design. Once the design has been validated, effort can be expended to engineer a purely computational implementation of the system.

Notably, this comparison can be usefully made at the design stage of creative system development, before developing complex computational modules. By first prototyping the system with ideal, human-delegated modules, its designers can evaluate the performance potential of its high-level design before investing effort in the expensive process of developing a fully computational system.

This new paradigm, which we call *ideal-module prototyping* or *IMP*, represents a powerful tool that is not currently available to the computational creativity research community. It gives researchers a method for validating the designs of new creative systems and for diagnosing flaws in existing creative systems by comparing them to an idealized version.

In this paper, we argue for the following position: *computational creativity research should be conducted using modular system design evaluated via ideal-module prototyping*. We describe ideal-module prototyping in detail, relate an experiment we conducted with applying IMP to an existing creative system, present arguments to defend our position, and discuss the impact that this paradigm could have on the future of computational creativity research.

Ideal-module Prototyping (IMP)

This prototyping paradigm is powerful but not overly complex. The main insight is that much can be learned about a modular creative system by replacing its computational modules with human-delegated versions that fulfill the same task or operation. In this section we detail the paradigm, and in the next we relate an experiment we conducted with it.

Note that although IMP is primarily intended for prototyping the design of a new creative system before the system itself is engineered, it can also be retroactively applied to evaluate and diagnose completed systems. The paradigm is applied similarly in either case, with the main difference being that the computer modules of an existing creative system are already well-defined and can be compared to directly.

Building Human-delegated Modules

When the goal of a creative system is to produce human-compatible output, the generation of that output often requires knowledge or computation that mimics human cognition to some degree. Modular creative systems have specialized modules that are called upon to perform these critical tasks in the creative process. Because these tasks attempt to operate at a human level, human cognition is naturally capable of completing them as well.

We note the obvious difference that exists between people's abilities to perform cognitive tasks relevant to the creative process, for example the difference between an expert and a layperson. We posit, however, that the difference in skill level between expert and layperson is much smaller than the difference between a layperson's skill and the skill of state-of-the-art computational modules for many tasks that are useful to the creative process. If the difference between the state-of-the-art module and a layperson's performance is small, then a human-delegated module is likely not worth developing for that task.

The first step in implementing ideal-module prototyping is to identify the system's modules and their inputs and outputs. Care should be taken to specify each human-delegated input and output exactly so that that information is as similar as possible to what the computer module processes. Each human-delegated module must be designed to be displayed and answered via a human-friendly UI. Care should be taken with any instructions issued to the human delegate to ensure that the human has neither more nor less information than the computer module when completing the task.

It is also important to account for differences between how people and computers answer questions, especially for modules that compute tasks with no input that reflects knowledge of larger system goals or previously computed tasks. When interacting with such modules, people will naturally remember previous task inputs which could influence their responses. For example, the story-writing system we experimented with presents human delegates with analogy completion tasks, the outputs of which are used as input for later analogy tasks. If one person were to fill in these successive analogies in sequence, memory of previous tasks could influence their answers in a way that deviates from the computer module's stateless operation. This effect can be mitigated by randomizing sequential tasks to reduce the likelihood that one person's responses will be influenced by memory.

Various approaches can be taken to designing a user interface that facilitates human participation in the creative system. For example, in our experiment with applying IMP to a story-writing system, we built a client-server web interface for participants to interact with. This design allowed for widespread deployment via the Internet to attract a diverse set of participants to the experiment.

In order to design user interfaces that accomplish the researchers' design goals, useful techniques may be drawn from the broader field of HCI. Lessons from Wizard-of-Oz techniques (Fiedler, Gabsdil, and Horacek 2004), which human-delegated modules may resemble from a UI or interaction perspective, and participatory design (Muller and

Druin 2007), which concerns inter-domain experiences, may be of particular use. We reiterate, however, that the goal of applying IMP is to validate the design of an ultimately computational creative system; human-delegated modules will be replaced by computational modules in the final system.

Once all human-delegated modules have been created, they can then be connected with the same high-level information flow that the system would use if it had computer-controlled modules. In that way, the complete human-delegated system computes the same algorithm as a purely computer-controlled version of the same system.

In the case that the human-delegated modules are designed to match an existing creative system, it is likely that the high-level information flow will need to be re-engineered to accommodate for the asynchronous or event-driven nature of human-computer interaction.

Finally, as the human-delegated systems implemented under this paradigm are intended for evaluation and diagnostics, the systems should include logging of useful data that is generated as the program runs. In addition to the final system output, any other data that will give system designers insight into the system's operation should be collected. Input and output data from each module are likely to be very useful for design analysis because they can be compared directly with corresponding data from computational versions of those modules.

Running the Human-delegated System

After the system is complete, the next step is to recruit people to participate in completing tasks for the system's modules. This may be approached in whatever way the researchers believe will yield the best results for their system. Commonly, the modules in a computationally creative system compute tasks that are relatively simple for humans to complete. This should permit the recruitment of participants from a wide range of backgrounds and education levels.

We note that concerns such as population size and statistical significance are likely of lesser importance when using IMP to evaluate the performance of a creative system. At the design stage, such evaluation is largely subjective and depends on the researchers' goals for their system. In this way, testing a system's design with human-delegated modules is similar to running a pilot study.

The number of participants to recruit depends mainly on estimating how many people it will take to provide a sufficient quantity of diverse module outputs to complete system execution in a reasonable time period. It is a near certainty that the human-delegated system will take more time to run than a purely computational system. However, we anticipate that with ideal, human-delegated modules a small number of complete program runs should be sufficient to collect useful data with which to evaluate the creative system.

Once the participants have been recruited and the creative system has finished delegating tasks to them, the results can be analyzed. With proper data collection, the results should give insight into the performance of the system's individual modules as well as its holistic performance.

The primary question is whether the creative system's output is satisfactory. Although this criterion wholly depends on the goals that the researchers have for the system, the data collected by this method should aid any assessment of whether the system achieves those goals.

If the output is satisfactory, then the researchers can move forward with engineering a purely computational version of the system, armed with the knowledge that their algorithmic design is valid. Alternatively, if the output is unsatisfactory, the more granular data collected during execution can be used to inform design improvements.

Data should be collected for each module in the system, for example by recording all inputs and outputs to each module during execution. This will allow for analysis of individual module performance, as well as diagnosis of how information flows between modules. Recall that although similar analysis could be carried out on a purely computational system, IMP features ideal modules that eliminate the conflation of flaw-causing behavior between imperfect modules and flawed information flow.

If the individual modules' inputs and outputs are found to be satisfactory, then unsatisfactory system output may be the result of flaws in the information flow between modules or require remediation via the modification, addition, or subtraction of modules. In the former case, it may be possible to redesign the connections between modules and simulate execution using the recorded input/output data, alleviating the need to recruit participants again. If that is not possible or if other modules are needed, partial reuse of the recorded data may still reduce the burden on participants.

It is possible that a given creative problem is not factorizable into modular tasks. Analysis of whether a given problem is factorizable or not is outside the scope of this work. However, we note that it may be useful to investigate the distinctions between problems that arise from an unfactorizable creative problem, the incorrect factorization of a factorizable problem, and software bugs in the implementations of a correctly factorized problem.

An alternative result of the evaluation is that one or more modules do not perform satisfactorily. If the modules' tasks are delegated to humans, their performance should represent a cognitive ideal. Thus, any performance failings in the modules that cannot be attributed to UI flaws warrant careful consideration from the researchers. Useful questions to ask may include what the purpose of the module is, why calculation of that operation is out of reach for humans, and whether it is tractable to build a computer module to compute something that humans are apparently incapable of. It may also be useful to consider whether an ideal computational module exists to complete the task, in which case that should be used instead of a human-delegated one in both the prototype and the final system.

The nuanced outcomes of evaluating a creative system in this manner show the power of ideal-module prototyping for teasing apart complex, interconnected interactions between modules and accurately diagnosing the origins of low-quality output.

IMP Checklist

To summarize IMP, we present a checklist of steps that should be taken to apply the paradigm to a creative system,

whether it is being designed or is already implemented. This checklist should be applicable to any modular creative system, and we encourage researchers to tailor its application to the needs of their system.

1. Identify computational modules in the system.
2. Implement human-delegated versions of all modules for which no ideal computational module exists, and engineer a version of the system that uses those modules.
3. Experiment with the human-delegated system by recruiting participants and having them complete the delegated tasks. Log useful data generated during the experiment.

The result of applying this paradigm is a set of experimental data that can be used in various ways to validate the system's design or identify shortcomings in the system's modules or information flow. Based on the results of the experiment, it may be useful to apply IMP again after modifying the system's design.

HIEROS Experiment

In this section, we give an example of how ideal-module prototyping can be applied to an existing creative system. It is instructive to consider such an example both as an exemplar of how to apply IMP and a concrete example of what can be learned from doing so.

HIEROS

HIEROS (Spendlove and Ventura 2020) is a computationally creative system that writes six-word stories, a genre of microfiction that exhibits several properties that make it interesting for study. Their short length reduces the amount of data that needs to be generated and allows for rapid human evaluation of results. Despite their brevity, however, they are far from a trivial creative domain; effective six-word stories push the limits of semantic and linguistic constructs to deliver impactful or emotional experiences to readers. Six-word stories are also not far removed from raw analogy and semantics. Because it does not contain filler or unnecessary words, the quality of a six-word story is closely related to the quality of the underlying relationships between the story's words.

HIEROS takes advantage of that final property to inform its modular system design. Two of HIEROS' three modules select the words that will form a six-word story, and the third assigns a score to the story that reflects its quality. The remainder of the HIEROS algorithm passes information between these same modules to search for higher-scoring stories. Consequently, the success of the system is very closely related to the quality of its modules, making it an ideal candidate for evaluating using our ideal-module paradigm.

Figure 1 shows a diagram of how information flows between HIEROS' modules. We will describe this information flow and each module in turn.

Using human-written exemplar six-word stories scraped from the web, HIEROS infers an underlying format for each story. This format includes a hierarchy graph of the words in the story computed using dependency parse information provided by the Stanford Parser (De Marneffe, MacCartney, and Manning 2006). In this acyclic hierarchy graph, each word is connected by an edge to a parent word—as determined by the dependency parse—except for a single root word that has no parent.

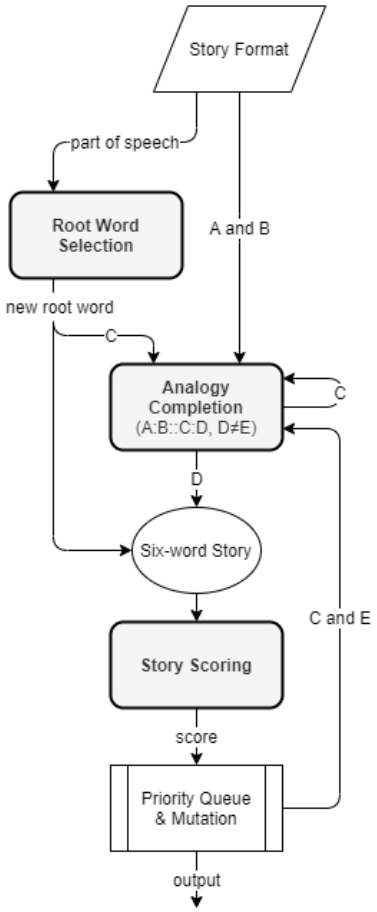

Figure 1: The flow of information through HIEROS' three modules (in bold): root word selection, analogy completion, and story scoring.

The relationship between each parent and child word is used to guide the selection of a new child word when presented with a new parent word. This task can be thought of as a classic analogy completion question $A : B :: C : ?$, with the original parent and child taking the place of A and B, respectively, and the new parent taking the place of C.

HIEROS uses an analogy completion module to accomplish this task. The inputs to the module are words A, B, and C. The module's output is a word D that fulfills the analogy $A : B :: C : D$ and shares the same part of speech as B. An optional input word E can be specified as a word that should be excluded for consideration as output.

HIEROS generates new stories by first selecting a human-written exemplar whose hierarchy graph will provide the format for generating a new story. It then selects a new root word via another module. The root word selection module simply takes a part-of-speech indicator as input and returns

a word of the appropriate part of speech on which to base the story.

Starting with the newly selected root word, the analogy completion module computes analogies for each edge between the root word and its children in the selected format's hierarchy graph. Then analogies are computed for those children's children and so on until all six words of the new story have been selected. The words in this new six-word story are not the same as the words in the original, but the new words share similar relationships between one another as the words in the original do, as guided by the format's hierarchy graph.

After generating a new story, HIEROS assigns it a score via a story scoring module. This module simply takes a story as input and returns a numerical score, for example a real number between 0 and 1. The module should assign high scores to stories of high quality and low scores to low-quality stories.

Once a story has been scored, it is placed into a priority queue, after which the highest-scoring story is dequeued and mutated to search for higher-scoring stories. A story is mutated by selecting a non-root node in its corresponding hierarchy graph, using its parent node to compute a new analogy for that word in the story via the analogy completion module, and recomputing analogies for all words lower in the hierarchy than the mutated word. The initial analogy recomputation provides the optional input word E to the analogy module to specify that the output should not match the pre-mutation word.

The mutated story is scored by the story scoring module and inserted into the priority queue, after which the highest-scoring story is dequeued and mutated, and the process repeats.

By generating, scoring, and mutating stories in this manner, HIEROS refines its stories, searching for progressively higher scoring stories. If a specified number of mutation steps have passed without finding a higher scoring story, execution terminates, and the highest scoring story seen during execution is returned as the system's creative output.

Thus, we completed step one of the IMP checklist. HIEROS' three modules, as represented in Figure 1, are:

1. An **analogy completion** module which takes words A, B, C, and an optional word E, and returns a word D such that $A : B :: C : D$ holds, $D \neq E$, and D has the same part of speech as B.
2. A **root word selection** module which takes as input a part-of-speech indicator and returns a word of that part of speech that is an interesting word on which to base a story.
3. A **story scoring** module which takes a six-word story as input and returns a numerical score in a certain range, with high scores being assigned to high-quality stories.

Note that each of these modules computes a task that is difficult to rigorously define. What does it mean for the relation $A : B :: C : D$ to hold between four words? What is the definition of "an interesting word on which to base a story"? What criteria should be used to assign a high or low score to a story? The nature of these difficult-to-define questions makes these modules both scientifically interesting and difficult to compute. HIEROS' computer modules that attempt to compute their respective tasks find some success but often fall short of ideal performance, and the system as a whole produces somewhat poor stories.

The key question, then, is whether low-quality output is the result of one or more flawed modules or is caused by a more systemic design issue. In order to more clearly diagnose HIEROS' shortcomings, and to gather useful data that could be used to improve HIEROS' design, we experimented with applying our ideal-module evaluation paradigm to the system.

Applying the Evaluation Paradigm

We re-engineered HIEROS to use human-delegated modules via a web browser UI and recruited participants to interact with it over the course of a week in order to test the system's performance with ideal modules.

We created a simple HTML and Javascript web page that asks the participant to complete delegated tasks corresponding to HIEROS' three modules: providing a root word, completing an analogy, or scoring a story. The web page connects to a server that runs the HIEROS algorithm and tells the web page which task to present to the user. Each task is used as module output in the system and advances the internal state as far as it can before requiring more tasks to be completed. After completing a task, the web page displays another task to the user, and users can complete as many or as few tasks as they like.

Each task is worded to provide clear instructions to the user without providing extra information that could be incorporated into the response. The tasks are designed to be completed by any person who is fluent in English and can understand and follow the instructions. Tasks are not explicitly randomized, but the server does not immediately give users subsequent tasks that are generated from the previous task they completed. This prevents a user from completing the entire process of creating and scoring a single story.

The root selection task prompts the user to "enter an interesting word to base a story on", including a desired part of speech.

The analogy completion task asks the user to "complete this analogy", followed by the analogy presented in "A:B::C:___" format, including specifying parts of speech. Neither the root word nor the analogy tasks enforce the part-of-speech specification; if the human user makes a mistake it is still considered valid input.

The scoring task presents a story and instructs the user to "score this story" with an integer score between 1 and 100, with no instructions pertaining to which aspects of the story to score. Integers were used instead of real numbers for ease-of-use.

Implementing these tasks as human-delegated modules constituted step two of the IMP checklist. Step three was to recruit participants and carry out the experiment.

To recruit participants, we distributed a link to the web page via social media and email. We did not collect demographic data. Users interacted with the web page over the course of a week, generating 142 stories total, including all

generated and mutated stories. To facilitate participant interaction, the human-delegated system did not terminate execution after a fixed number of mutation steps with no score increase, in contrast to the original HIEROS system.

Results

Examining the results of the human-delegated system, we find that the system's stories are satisfactorily coherent, diverse, and interesting overall. This result provides strong evidence that HIEROS' design is sound. Furthermore, comparing the results of the human-delegated system to the original computational system yields insights into which parts of the original system could be improved.

Although only the highest scoring story would be returned as its output, it is instructive to inspect all the stories the human-delegated system generated. The nature of the system's scoring and mutation algorithm results in many stories that differ only by a few words. Filtering out very similar stories, the top four scoring stories from the human-delegated execution are:

Wilted. I Was a recent past.

Lonely outcast. Overgrown wildernesses. Farewell society.

Brave outcast. Horror yearns. Expel society.

Super chewy. Dry Food. Creature Meal.

The first story was created using the format of the web-scraped exemplar "Shit. I AM the adult supervision." and demonstrates an interesting difference in subject matter while still adhering to similar inter-word relationships. The second and third stories also differ intelligently from their shared exemplar "Bitter taste. Swollen lips. Bye lover." and are examples of two mutations of similar stories. The fourth story comes from the exemplar "Last human. Wrong Planet. Alien Delicacy." and demonstrates how the food-related analogies present in the original influence the direction the new story takes after starting with the new root word "creature".

Conversely, stories written by the original HIEROS system are less coherent and tend to represent unintelligent synonym substitutions for most words in the exemplar story. Examples follow, including the exemplar story in parenthesis for comparison:

Cried ourselves helplessly. Sobbed them asleep. (Cried myself asleep. Screamed myself awake.)

Ought Myself certainly rent one bullet? (Can I just buy one bullet?)

There gets unending voluptuousness from fecundity. (There is immense beauty in diversity.)

Be possibility to anybody that lives. (Be kind to everything that lives.)

Despite the low quality of the original system's stories, the ideal system's results demonstrate that HIEROS' story writing algorithm could be a viable method for writing interesting and novel six-word stories. The only difference between the two systems is their modules, so it follows that the difference between their stories' quality can be accounted for in the difference between the ideal modules and the computer modules.

The primary module that guides HIEROS' creation of six-word stories is the analogy completer, which answers analogy questions of the form $A : B :: C : ?$. The computer module that HIEROS uses to solve these tasks is structured as follows.

A semantic vector is calculated for words A and B (the parent and child words from the hierarchy graph, respectively) by subtracting their word2vec word embeddings (Mikolov et al. 2013), effectively encoding the semantic relationship between the words. That semantic vector can then be added to the embedding of a new parent word C to produce a new child word D that shares the same semantic relationship, i.e. it satisfies the analogy $A : B :: C : D$. Word D is then returned as the module's output.

Examining the original system's output, we observe that the computer analogy completer often simply returns synonyms of B (the word in the original story to be replaced) as its output. The module does not seem to account for the relationship between A and B and how it could be applied to C, even though word2vec's embeddings are often touted as facilitating analogical reasoning via simple geometric operations.

Meanwhile, the input and output data we collected from the human analogy completion module demonstrates a more nuanced and intelligent selection of analogous words. Three groups of example analogies demonstrate patterns in how participants completed this task.

The first group are analogies that cleverly apply an aspect of the relation between A and B to the word C to derive an output word D. Examples of this group include:

taste : bitter :: outcast : lonely

lips : swollen :: wildernesses : overgrown

smiles : someone :: rots : tree

A second group are analogies in which the left-hand side is not interesting or evocative. In this case, participants often fell back onto choosing a synonym of B, as displayed in these examples:

tempted : twice :: expired : once

luck : still :: tears : seldom

Another interesting group of responses seems to occur when there is a meaningful relationship between A and B, but not one that can be easily applied to C. In this case, participants chose words that were primarily related to C:

smiles : today :: eat : bread

supervision : adult :: past : history

went : anyway :: sought : fearlessly

All the data collected via this experimentation method is open to interpretation and is intended to aid the researcher in evaluating and improving their own system. As such, we do not make any strong, statistical claims about the results, we merely report our subjective observations of useful trends.

HIEROS' scoring module is of critical importance to the system's overall performance. Even if the system's story

generation is somewhat flawed, an accurate scoring module could still guide the system to high-quality output.

HIEROS' scoring module succeeds in assigning low scores to low-quality stories but is unable to consistently assign high scores to high-quality stories. This results in a large variance between scores assigned to generated stories, confusing the system's ability to search the space of similar stories for higher quality stories.

The data collected from the human-delegated scoring module reveals that participants were not as consistent in their scoring as they were with analogy completion. Scoring stories on a numeric scale with no guidance or context is a difficult task for humans, and it was presented without modification in order to mirror the task that the computer module computed.

Quality differences between two stories are highly subjective, making it difficult to analyze small differences between human-assigned scores. Examination of broader trends in the human-assigned scores, however, seems to indicate that high-quality stories are indeed assigned higher scores than low-quality stories. Compared to the previously presented highest-scoring stories, the following lowest-scoring stories demonstrate a markedly lower quality level:

Lonely outcast. Benjy yearns. Expel society.

Swallow the biscuit tree rots eternity.

Wilted. History Was a zookeeper penguin.

Aside from the expected result that humans are accurate story scorers, a more nuanced insight emerges from examining patterns in the stories generated by the human-delegated system.

Because the HIEROS algorithm selects stories to mutate based on score, the system generated many stories that were mutations of the same story. When a new story was introduced, its first generation was often not of the same quality as the current high-scoring story which had been iterated and improved upon. This resulted in new stories never having a chance to be refined because the system ignored them in favor of the reigning champion.

This suggests that a division be made between scored stories as they compete to be mutated. Instead of a single priority queue organized by score, the system would likely be improved by keeping a separate list of newly generated stories from which stories are selected at random to mutate and score. This would give newly-generated stories a number of iterations to be refined before being discarded.

HIEROS could still maintain a priority queue but only use it for refined stories whose scores surpass a certain threshold. This more nuanced refinement algorithm should prevent the priority queue from being dominated by already-refined stories that exclude new stories and deny them the opportunity to be refined.

We note that this insight was only made possible by the ideal module experiment; HIEROS' original scoring module was not skilled enough to make such fine distinctions. By running the system with ideal, human-delegated modules we were able to gather unique information that informs a better system design.

A New Prototyping Paradigm

Prototyping is an invaluable tool in any design discipline (Thomke and Nimgade 2000; Gerber 2010). It allows designers to evaluate a design while it is still in the early stages of development and avoid expending time and resources on implementing a design that will ultimately prove unsatisfactory.

Computational creativity is a challenging field that requires the design of complex systems. We argue that ideal-module prototyping represents a powerful method for evaluating designs of new and existing creative systems and that adoption of this paradigm will benefit the computational creativity community.

Improving Creative System Design

Ideal-module evaluation can aid the improvement of a wide variety of creative systems in different ways depending on the nature of their designs. Systems that attempt to improve the results of an already functioning creative system, completed systems with unsatisfactory results, and entirely novel systems can all benefit from applying this design evaluation paradigm.

Meta-analysis of computational creativity research, such as that conducted by Colton and Wiggins (2012), reveals recurring patterns in the development of creative systems. Colton and Wiggins describe one such observation that they call the *latent heat effect*, which describes the phenomenon that "as the creative responsibility given to systems increases, the value of its output does not (initially) increase". As increasing the creative responsibility of computational systems is an implicit goal of research in this field, it behooves us to better understand and reason about this phenomenon.

Ideal-module prototyping provides a tool with which to probe and validate system designs, and applying it to designs that take on increased creative responsibility will allow researchers to better diagnose the causes of the latent heat effect. Increasing such responsibilities likely involves the addition of new modules to a system or a significant modification to the system's information flow. Applying IMP to the newly designed system will allow the designers to verify that those changes did not invalidate the system's capability to generate high-quality output when operating with ideal modules.

Thus, this new paradigm can help explain the latent heat effect and provide a strong argument that although a system with increased responsibility currently produces inferior output, it has the potential to generate high-quality output in the future.

We anticipate that IMP evaluation will be most beneficial when it is applied early in the design of a creative system. Embarking on the development of a new system requires trust that the design will prove fruitful in the end. Experimenting with an ideal-module prototype before investing development effort into building complex new computational modules allows researchers to determine ahead of time whether it will be worth expending that effort.

Building ideal, human-delegated modules requires precise definitions of those modules' interfaces. This forces de-

signers to carefully define the exact responsibilities of those modules early in design, an exercise that is beneficial to system designers in its own right.

In addition to saving time by alerting researchers to unproductive design routes early, careful application of this paradigm could result in a system in which human-delegated modules can be swapped in place for computational versions without requiring changes to the higher-level control logic, thereby reducing development time.

Although IMP is primarily intended to be applied early in design, existing creative systems with lackluster output can benefit from its application as well. As detailed in our experiment with HIEROS, this method can provide more exact insight into why a system is failing and give its designers data that they can use to improve its design. We are hopeful that by accurately diagnosing their flaws and pointing the way toward how to improve them, the introduction of this paradigm will aid in rescuing unsatisfactory creative systems that have previously been shelved.

Collaboration & Modular Systems

Beyond application to single creative systems, we anticipate that adopting IMP will have a positive effect on the computational creativity research community as a whole.

The requisite thresholds of trust and confidence required to invest resources into developing a creative system are multiplied when dealing with collaborative research projects. IMP provides a means by which such confidence can be established before serious collaborative investment is made. Researchers seeking collaboration could first evaluate their design via this method to demonstrate its validity. Alternatively, this method opens up opportunities for useful replication or analysis of previously presented creative systems.

Modular designs lend tractability and flexibility to computationally creative systems. Researchers in the computational creativity community should design modular creative systems with precisely defined module interfaces and responsibilities. In addition to simply being better design practice, this will allow researchers to take advantage of the power of applying ideal-module prototyping to their designs.

Modular designs also present unique opportunities for collaboration. If multiple creative systems are designed to use an identical module, any effort spent developing and refining that module will pay off multiple times. Existing computational modules such as word2vec, WordNet (Miller 1995), and GPT-2 (Radford et al. 2019) are examples of modules that are widely used by different systems. By employing this ideal-module prototyping to validate shared-module systems before developing such computational modules, all involved parties can have assurance that their efforts will not be in vain.

Conclusion

The computational creativity community would benefit from a robust prototyping tool that allows creative system designs to be validated before putting in the challenging effort to implement them computationally. We have presented ideal-module prototyping (IMP), an evaluation paradigm for modular systems that compares imperfect computational modules with ideal human-delegated versions.

IMP can be applied to new system designs or retroactively to existing systems, as we demonstrated with our experiments with HIEROS. The results of the human-delegated system allow researchers to make strong claims about the validity of their system designs or accurately diagnose flaws in existing modular systems.

Researchers in the computational creativity community should design modular creative systems with well-defined module interfaces and employ ideal-module prototyping to validate their designs. Adopting IMP will allow for the improvement of existing creative systems, greater confidence in the development of new systems, and increased opportunities for collaboration.

References

Colton, S., and Wiggins, G. A. 2012. Computational creativity: The final frontier? In *Proceedings of the European Conference on Artificial Intelligence*, 21–26.

De Marneffe, M.-C.; MacCartney, B.; and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation*, volume 6, 449–454.

Ehsan, U., and Riedl, M. O. 2020. Human-centered Explainable AI: Towards a Reflective Sociotechnical Approach. *arXiv e-prints* arXiv:2002.01092.

Fiedler, A.; Gabsdil, M.; and Horacek, H. 2004. A tool for supporting progressive refinement of wizard-of-oz experiments in natural language. In *International conference on intelligent tutoring systems*, 325–335. Springer.

Gerber, E. 2010. Prototyping practice in context: the psychological experience in a high tech firm. *Journal of Design Studies*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv* abs/1301.3781.

Miller, G. A. 1995. WordNet: A lexical database for English. *Communications of the Association for Computing Machinery* 38(11):39–41.

Muller, M. J., and Druin, A. 2007. Participatory design: the third space in hci. In *The human-computer interaction handbook*. CRC press. 1087–1108.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners.

Spendlove, B., and Ventura, D. 2020. Creating six-word stories via inferred linguistic and semantic formats. In *Proceedings of the 11th International Conference on Computational Creativity*, under review.

Thomke, S., and Nimgade, A. 2000. *IDEO product development*. Harvard Business School Cambridge, MA.