2010

Sustainable printing activities: design and initial approach of a print energy life-cycle decision tool

Elvis Montero

J. Scott Hawker

Marcos Esterman

Sandra Rothenberg

Sandra Rothenberg

Sustainable Printing Activities: Design and Initial Approach of a Print Energy Life-cycle Decision Tool

By

Elvis Montero
Graduate Student, Information Technology, Golisano College of Computing and Information Sciences

J. Scott Hawker, Ph.D.
Assistant Professor, Software Engineering, Golisano College of Computing and Information Sciences

Marcos Esterman, Ph.D.
Assistant Professor, Industrial & Systems Engineering, Gleason College of Engineering

Sandra Rothenberg, Ph.D.
Associate Professor & Zutes Faculty Fellow, Saunders College of Business

Rochester Institute of Technology

A Research Monograph of the

Printing Industry Center at RIT

No. PICRM-2010-03

R·I·T Printing Industry Center
An Alfred P. Sloan Foundation Center

Sustainable Printing Activities: Design and Initial Approach of a Print Energy Life-cycle Decision Tool

By

Elvis Montero
Graduate Student, Information Technology, Golisano College of Computing and Information Sciences

J. Scott Hawker, Ph.D.
Assistant Professor of Software Engineering, Golisano College of Computing and Information Sciences

Marcos Esterman, Ph.D.
Assistant Professor, Industrial & Systems Engineering, Gleason College of Engineering

Sandra Rothenberg, Ph.D.
Associate Professor & Zutes Faculty Fellow, Saunders College of Business

Rochester Institute of Technology

A Research Monograph of the
Printing Industry Center at RIT
Rochester, NY
January 2010

PICRM-2010-03

The research agenda of the Printing Industry Center at RIT and the publication of research findings are supported by the following organizations:

Table of Contents

Abstract

Information technology holds tremendous potential to help consumers and firms make more sustainable choices by providing information at key decision points. As one example, there are a number of software programs that help calculate and summarize environmental metrics for various products and processes. Surprisingly, while many printers are moving into the IT arena, the technology has not been fully utilized. For the most part, there is a lack of knowledge on the part of the consumer on the sustainability impacts of their communication decisions. Thus, this paper outlines a decision tool, presented to the consumer as they make a print decision, which estimates the energy consumption of printing a given document by analyzing the user's requirements for the print job, the printer selected and the corresponding life-cycle criteria for these elements.

Introduction

The environmental impacts associated with the entire print value chain include, but are not limited to, deforestation, emissions, water consumption, solid waste production, energy consumption, and air pollution. There has been growing interest and activity by the print industry to address these issues by understanding and reducing these impacts through prudent design and process decisions (Carli, 2007; Chadwick, 2008; Cross, 2008; Kemper, 2008). However, in order to effectively reduce these impacts, it is important that the appropriate metrics, methods and tools be available to enable good decision-making. A consistent need expressed by the print industry is the development of standardized sustainability assessments that would allow the comparison of different printing technologies, printing platforms, printing products and printing value chains.

This paper is part of a larger research effort by the Sustainable Print Systems Laboratory (SPSL). The goal of this larger research effort is two-fold (1) to understand and characterize the metrics and methods employed by the printing industry to measure, track and integrate sustainability into their business practices; (2) to develop methods and tools to aid the print industry to become more sustainable. The focus of this monograph is the latter, in which the preliminary design of a print energy life-cycle decision tool is detailed as prototype for other life-cycle decisions tools. Information technology (IT) holds tremendous potential to help consumers and firms make more sustainable choices by providing information at key decision points. There are a number of software programs that help to calculate and summarize environmental metrics for products and processes. While many printers are integrating IT into their processes, this technology has not been fully utilized to improve sustainability. In particular, there is an opportunity to better inform the consumer of the sustainability impacts of their communication decisions.

The need for a print life-cycle decision tool that provides users with information on the impacts of their print choices has been identified by several authors (Curtin, 2008; Jones, 2008). Print providers have recognized this, and have started to provide some tools to consumers, with a focus on carbon (See Appendix A). While a variety of tools exist, there are some limitations. First, the reported outcomes tend to report averages and do not include factors such as the location and environment of the user. Second, the requirements and characteristics of the documents to be printed and their final purpose are not factored in the decision. Lastly, the information is not integrated in any way into the print decision itself.

Figure 1 presents our vision for a print life-cycle decision tool that addresses these limitations, in which the user would be presented with information on the impact of a print job from a number of perspectives. Using RIT printers as a test bed, this IT tool would analyze the impact of a document before it is printed; namely, the program would examine the alternatives of printing the document, any possible alternatives to this action and the end result in economic, quality and environmental terms.

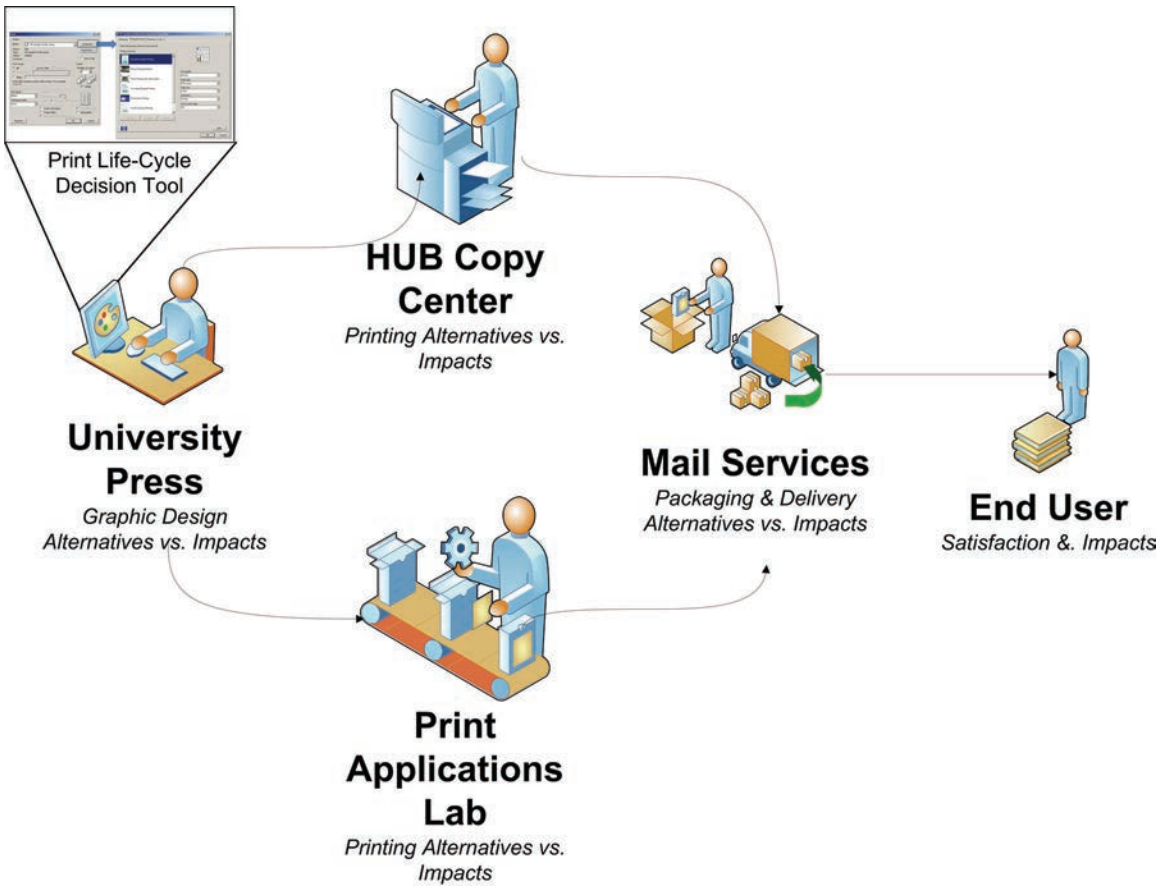


Figure 1: Print life-cycle decision tool test bed

The test bed proposed in Figure 1 poses a significant challenge and encompasses a large-scale research project. This is a significant undertaking, as it would involve understanding the Life Cycle Impacts (some of which have not even been determined) of a variety of printer technologies and devices, communication media, and transportation options, linking this information to other performance data, and developing IT systems to integrate this information. Thus, as a first step it was decided to focus on a small portion of this project: the measurement and integration of energy/power consumption into the print decision. After the direct associated costs of printing, energy is the sustainability metric that is perhaps of the most interest to the average consumer since it is easily translatable into money and a potential source of greenhouse gas emissions. As noted by one industry expert (Weil, 2008):

> "If your users don't care about how much money the new approach to print management is going to save the company (after all, it's not the like savings are going to show up in their paychecks), you might want to appeal to their inner tree-hugger."

Focusing on energy/power consumption as a first step towards a print life-cycle decision-tool is consistent with the ultimate goal of allowing print decision makers to evalu-

ate the trade-offs between economic, performance and sustainability metrics for each print decision.

System Requirements

Given the constraints above, it was decided that the system should provide the capabilities summarized below. (Note that the focus of this work is on the development of a proof-of-concept. Many of the details and specific numbers will need further development and validation. The basic design and software architecture is the main focus of this work.)

- Provide a way for the user to enter the following print request attributes:
  - Document reference (i.e. file to print)
  - The desired printer from the devices registered on the system
  - Number of copies
  - Pages (all or a range)
  - Media type
  - Print quality
  - Document color
  - Desired print quality (e.g. draft mode, paper selection, etc.)
- Provide an energy model of supported printer(s).
- Provide a way for the user to modify print request attributes and request new energy computations.
- Provide a way for the user to create an energy consumption report.
- Designed in such a way that allows it to be expandable.

User Interface

There are three different views associated with the user interface: job creation, historical reporting, and the energy model details.

Job Creation

The main idea behind this new tool is that upon initiating a print job, a revised user interface – a print screen (it is named "Powergy") – would appear. The user would specify a document; in this case either a Word document or a PDF file, in order to obtain an estimate of the energy consumed printing it. Figure 2 shows the user interface after a Word file has been selected. It displays the energy consumption for the printer at hand,

the number of pages in the input document, and the time it will take for the printer to finish printing. If the characteristics of the print job change, so does the energy used; the energy for each item in the display is computed dynamically based on the printer selected, the document and job values.

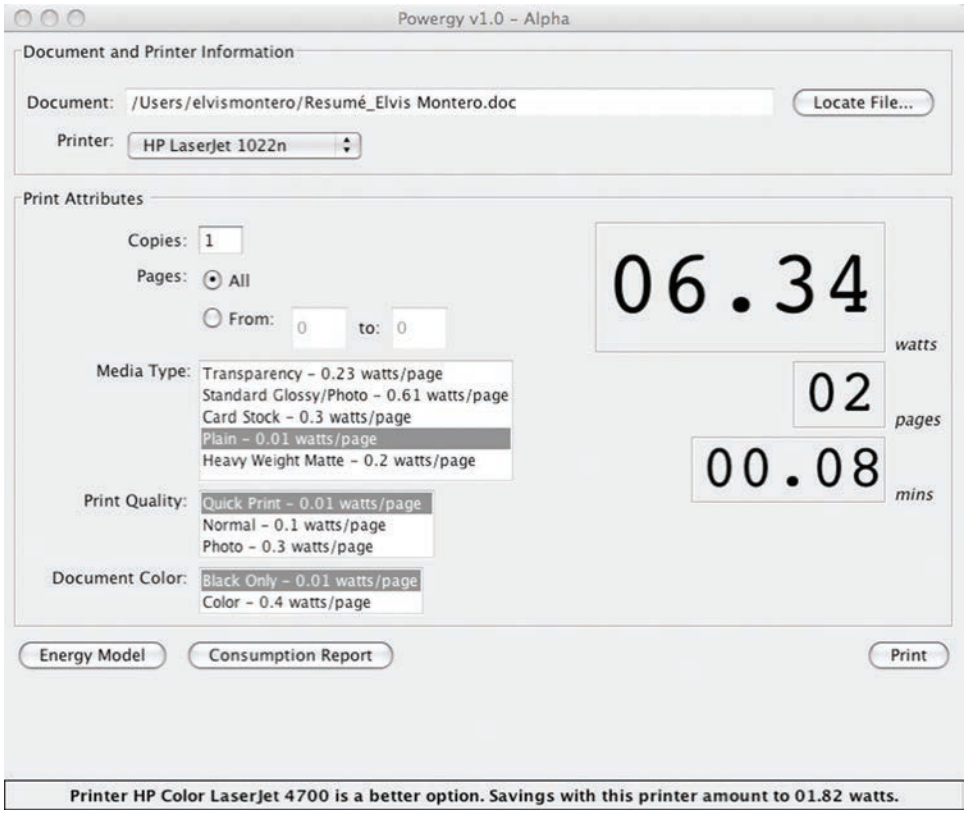


Figure 2: Powergy for a printed Word document

Historical Reporting

Figure 3 shows Powergy displaying historical data (i.e. consumption report in the last 10 days of usage). The report shows the aggregated energy consumption for the last 10 days with any activity registered. In other words, Powergy saves the energy consumption estimate when end users print the specified document.

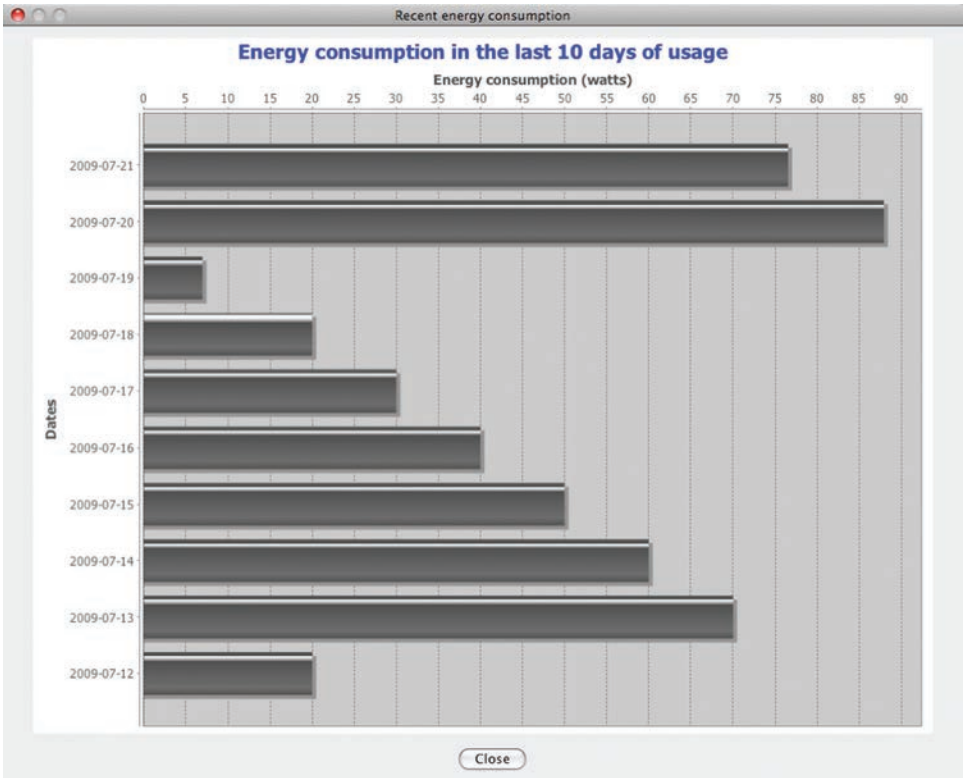


Figure 3: Powergy reporting historical energy consumption

Energy Model

This screen in not intended for the ultimate user of the tool, but instead it is intended for developers. In the future, it is envisioned that the energy model would be available in a variety of ways that range from complex computational models to the simple data entry interface shown in Figure 4. Through this simple interface, users can see how the energy use is being calculated and a system administrator can enter or modify the life-cycle energy values for new or existing printers, media types, and operational policies.

For the purposes of this demonstration, the energy consumption of the media production was lumped in with the printer energy model. In reality, it would have to be separated from the printer energy model and have an energy model of its own. Furthermore, the authors researched publicly available data and used this data, normalized per page, to populate the energy model. These are the numbers you see in Figures 2-5. These numbers assumed a constant use level of each printer. However, these numbers would be sensitive to printer usage and sleep mode behavior. Future efforts would have to obtain energy draw of a particular printer and then integrate overhead energy use.

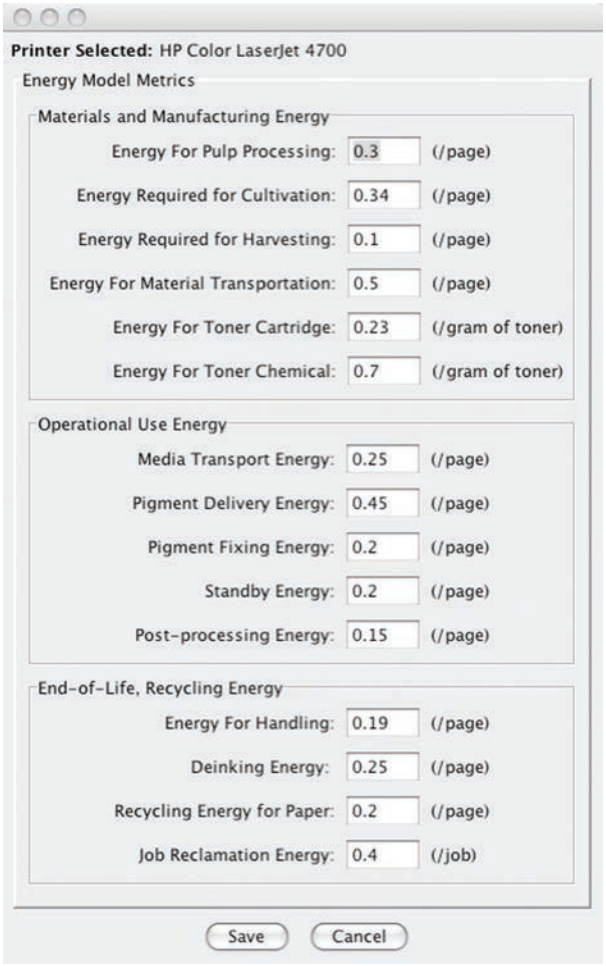


Figure 4: The elements from the energy model of Figure 2 used to compute the energy estimate

Development of Powergy

Powergy allows end users to specify the document to be printed, adjust the print job attributes, view the energy estimate for the document and attributes selected, browse recent energy consumption and configure the energy model.

Energy Model Abstraction

Before exploring the solution's design and implementation, it is necessary to discuss the energy model abstraction. In order to provide a sound estimate of the energy consumption of a print job, an energy model is required that enumerates what elements must be taken into account prior to running the computations. The energy consumed when a document is printed is more than just the printer's active power consumption. In order to ground this discussion in an actual example, Table 1 shows the main elements that were used in the development of this prototype. It was decided to break the energy model down into two dimensions. One dimension handles a typical life-cycle breakdown, where the energy consumption associated with various life-cycle stages needs to be accounted for. The second dimension that was used was a breakdown as a function of print value chain agents. This is not the only way to decompose the model, but for our purposes this provided a useful breakdown.

Table 1: Energy model dimensional elements

		Value Chain Agent				
		…	Device	Print Job	Document	Page
Life-Cycle	Materials Extraction					Forestry
	Materials Processing					Pulp Processing
	Manufacture					
	Use					
	End of Life					

This breakdown allowed for the tracking of energy consumption at various levels of aggregation and abstraction. For example, at the lowest level, there are page-level attributes that are of interest to track. Coupling this abstraction with the life-cycle dimension, it is now possible to isolate the elements that consume energy during the harvesting of the trees, the processing of the trees, the manufacture of paper, the transport energy while it is being printed on, etc. Similarly, this same kind of energy accounting can be done at the document-level, the job-level, the device-level, and other yet to be defined abstractions. As an example of the yet-to-be-defined category, there are activities that are associated with the order fulfillment process of the entire job that might include accumulation and packaging of all of the printed documents, delivery and potentially storage and retrieval. All of these activities would contribute to the consumption of energy. The goal is to have an implementation architecture that is flexible enough to handle not only the current metrics, but yet-to-be-defined metrics as well.

With this abstraction in mind, Powergy uses a software abstraction that sums the multiple sources of energy that reside within a single entity. That entity could be a single number, an equation or a more complex relationship composed of an aggregation of entities. The software design pattern used to abstract this concept is called the composite pattern (Lasater, 2007). The composite pattern allows us to deal with collections of objects and single objects as if they had consistent behavior. This is accomplished by allowing energy metric elements and aggregations of energy metric elements to be derived from the same base class. Lasater (2007) gives us a nice explanation of the idea behind this design pattern:

> "If you called the method on a leaf [single energy metric], then you only execute the code in that leaf class. But, in the case of the composite class [an aggregate of energy metrics], when you call the method, the composite loops through all the leaf objects inside the composite and calls the method on each one. This allows you to use either a single object's method(s) or get a compounded effect for a whole collection of classes by calling all the leaf classes in the collection."

Figure 5 illustrates the pattern as implemented in the code (the application's complete class diagram can be found in Appendix B):

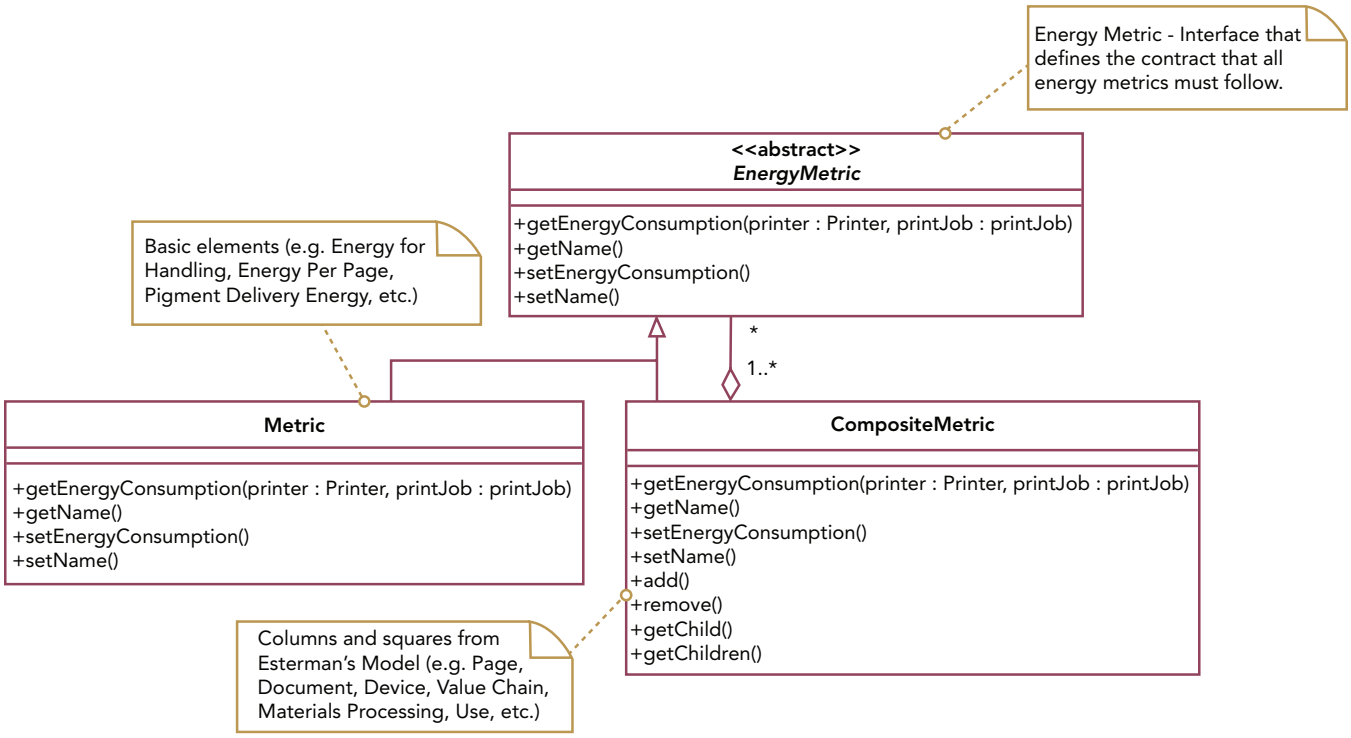


Figure 5: Composite pattern's implementation

Architectural Modules

With the discussion from the previous section in mind, this section will discuss the architectural modules needed to calculate the energy consumption, as shown in Figure 6. Each component will be briefly discussed below.

Adviser

The adviser serves two functions. It is the user interface that allows the user to interact with the application and enter the appropriate data. Its second function is to orchestrate the energy calculation and act on the resulting calculation. In this particular implementation, the adviser is simply a reporting mechanism. However, in the future it could just as easily make recommendations to the user based on embedded algorithms. Taking that a step further, automated actions and controls could be taken to optimize the performance metric of interest. While these actions will not be possible in the immediate future, the structure is in place to allow it to happen.

Energy Model

The energy model was discussed in quite some detail in the previous sections. What will be highlighted here is that the energy model has been separated from the device and value chain and is itself a stand-alone component. The reason for doing this is that it will allow flexibility in the future for the advisor to perform more sophisticated energy calculations. For example, one could envision that there is an embedded energy model in the device that has been provided by the device manufacturer. It is also feasible that a third party has also developed an energy model for the same device. The advisor could rely on one or the other or perform a calculation based on crosschecking between the models. The basic idea is that those calculation details would reside within the abstraction developed in the previous section and could be easily carried out irrespective of the source of the data.

Printer, Print Job, Document

These are all separate abstractions that contain data that would feed the energy model. For example, the printer could report out information regarding the amount of time that it has been idle, the state of the device (toner remaining, life of components, etc.), all of which would make for more accurate estimates. From the document one would derive the amount of pigment materials needed, the number and types of media. These are representative of state variables.

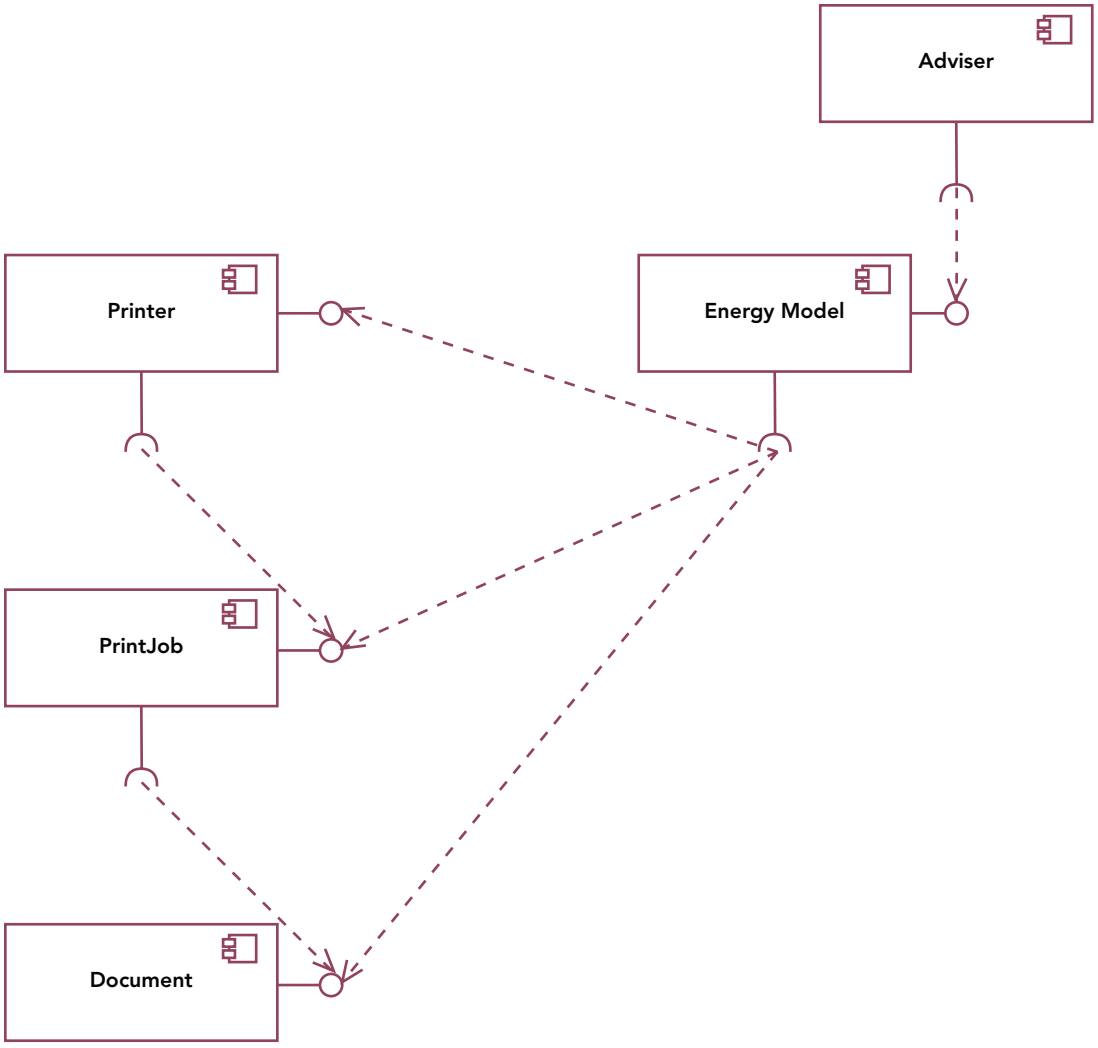


Figure 6: Component diagram presenting Powergy's architectural view*

To recap Figure 6, the adviser queries the energy model in order to determine the energy consumption for the document and printer selected at runtime. The energy model would, in turn, gather the information it needs from the printer, the print job and the document before producing an estimate. Thus, the energy model is parameterized by the specific printer, job, and document. The energy values are not merely "numbers." They may appear as a single number when the estimate is produced (such as multiplying per page energy costs by the number of pages in the print job). However, each metric handles how to generate the energy estimate for itself differently, as well as what information from the printer, print job and document are needed in order to produce the estimate.

*The interrelationships between the components underline the dependencies necessary to estimate energy consumption.

A solution such as the one outlined above is beneficial for several reasons. First, it increases independence between components. A printer does not need to specify its energy model. Similarly, a print job need not know about a printer, and a document need not know about a print job. This low coupling means changes in one part of the system minimally impacts other components. Second, as a direct consequence of this separation, the resulting code is easier to maintain and extend, thereby decreasing the time required to update the application. Third, a developer can easily add new energy metrics, modify their computations, and add new aggregations. Further, an end user can easily modify the energy model values for new printers. Last, this approach makes the code more likely to have functional cohesion (i.e. logically related parts of a module are grouped together since they all make for a single, well-defined task), which is a highly desirable trait to have in any software.

Other Technical Considerations

Figure 6 provided a high-level view of the design. The following sections cover the relevant details including the file formats that are supported by Powergy, the nature of the relationship between a printer, a print job and the energy model, as well as how the application is conducting printing.

Printing and Java

The Java Print API is Sun Microsystems' set of classes and interfaces designed to help developers handle printing for Java applications. With the help of the Java Print API, Powergy is able to create print jobs and send them to the chosen printer. So that the reader can get a sense of the kinds of attributes that the Java Printing API can make available to Java developers, Appendix C shows the output of a small program that lists all of the attributes found dynamically (using reflection) for one of the printers available on our system.

Unfortunately, the nature of what's available for each printer varies greatly depending on the device's manufacturer. Because of this situation, Powergy is only using a subset of the most commonly available options: *media type, print quality* and *color*. Even these basic elements are sometimes not found for a given printer (the detailed reader would have noticed by now these attributes aren't in the output pasted in Appendix C). In those circumstances, Powergy just sends the document to the printer without specifying the missing attributes.

The relationships among the *Printer*, *Document* and *PrintJob* interfaces are also relevant. With the help of the Java Print API, the *Printer* interface is effectively a proxy for any printer. Also, the *Printer* interface has a public *print* method that accepts a *PrintJob* object as its only parameter. In turn, the *PrintJob* interface has a reference to the *Document* that's going to be printed, as well as the attributes defined above (i.e. media type, print quality and color). Here's a snippet that shows an example of how to print using this approach:

```
printerInstance.print(printJobInstance);
```

Concrete classes that implement the Printer interfaces encapsulate the calls to the Java Print API. Figure 7 shows the class diagram describing the relationship between printer, print job and document (the complete class diagram can be found in Appendix B).

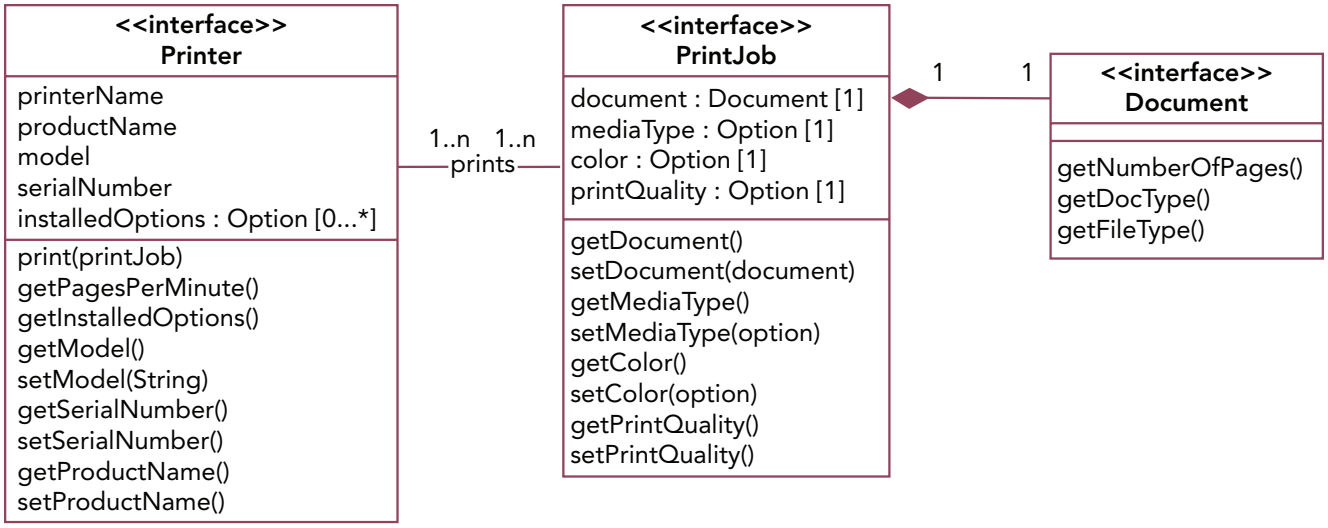


Figure 7: Class diagram showing the relationship between the Printer, Printjob and Document interfaces

Local and network printers are treated identically. As a matter of fact, Powergy doesn't care whether the printer is sitting right next to a user's computer or at a remote, distant location. If the printer is registered on the target operating system's printing system, Powergy will have access to it and thus, will also be able to send print jobs to it. If a printer is not on/online, Powergy would display an error message stating the document cannot be printed because the printer is not on or reachable at the time.

Powergy is not taking into account printer state beyond what the underlying operating system makes available for the application (i.e. whether the printer is on or off). In order to account for special printer state, e.g. warm up time, the Printer interface and the energy model must be updated so that this information is retrieved from detected printers and provided to the energy model before estimating energy consumption. Given the information that's available using the Java Print API, this is not a trivial task.

Documents

An element of paramount importance when estimating energy consumption for a given document is the number of pages. There is one caveat nonetheless. Different file formats require specific handling depending on the file type. A JPEG and a PDF file would need different parsers and thus, different handling code in order to obtain the number of pages for each case.

Since every file format supported would need special parsing techniques, thereby adding complexity with each new type, Powergy is initially supporting only PDF and

Word documents. Figure 8 illustrates the inheritance relationship between Document and its two concrete types (PDF and Word documents).

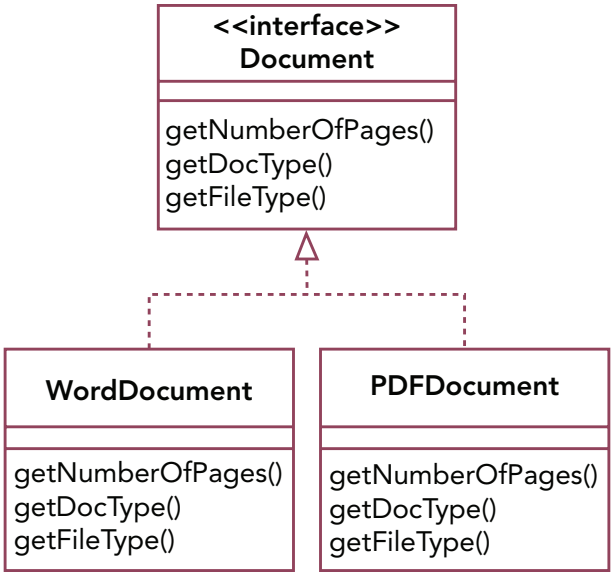


Figure 8: Document class diagram

The *getNumberOfPages* method is responsible for implementing the code required to parse the given document and return its number of pages. Powergy is using two external libraries for the two document types (one used for parsing PDF documents, and the other for analyzing Word files). Details of both libraries can be found in the External Libraries section.

If the reader wanted Powergy to support a different file format, implementing the Document interface for the new type would be necessary. Also, in order to have documents' content taken into account, the Document interface and the energy model would need to be updated. A concrete realization of a Document interface (e.g. PDF, Word document, TXT file, etc.) would have to inform the energy model about its content. In turn, a new metric must be created with the sole purpose of using the Document's content and the Printer object to estimate energy consumption. This new metric's value would then be aggregated together with the overall estimate of the energy model.

Reporting Energy Consumption

Powergy generates a report of the energy consumption in the last ten days of usage. The purpose of such a report is to give the end users an at-a-glance view of their recent energy consumption patterns. From Powergy's viewpoint, the energy estimate is saved when the end user actually prints a document successfully. When that happens, the estimate is added to the day's total and the new figure is presented.

Since reporting is not Powergy's main goal, the report generated is only included as an illustration of the reporting capabilities that a more advanced print energy life cycle decision tool could incorporate. As a result of this fact, there aren't any transaction boundaries, or complex integrity checks in the code. Figure 9 shows the table used to save the data after the print job has been sent to the selected printer.

daily_report	
PK	**date**
	energy_consumption

Figure 9: Table used by Powergy to save historical energy consumption data

The following SQL commands are sent to the database system during creation, selection and modification of the data respectively (the question mark denotes a parameter being passed):

```
CREATE TABLE daily_report (date DATE NOT NULL , energy_consumption DOU-
BLE NOT NULL , PRIMARY KEY (date))

SELECT * FROM daily_report WHERE date = ?

INSERT INTO last_10_days_report(date, energy_consumption) VALUES(?,?)

UPDATE last_10_days_report SET energy_consumption = ? WHERE date = ?
```

If the end user does not print anything on a given day, Powergy would display a bar with length zero for the holder of that particular day.

External Libraries

Plotting the consumption report chart and parsing Word/PDF documents are tasks handled by external libraries. The libraries below are used in the application to deliver specific, special functionality.

JFreeChart: Allows Java developers to easily generate charts. The energy consumption report, a horizontal bar chart, was generated using JFreeChart's library. More information about the library can be found at: **http://www.jfree.org/jfreechart/**

Derby: A relational database management system. Powergy uses Derby to store energy estimates for submitted print jobs. What differentiates Derby from other RDMS is that the database engine itself can be run as part of the invoking application (i.e. within the same Java Virtual Machine instance running the application) in what's called *embedded mode.* In embedded mode, the application can read and write using SQL commands

to Derby files in the hard disc. This reduces the application's memory footprint while allowing the use of a flexible DB engine. Apache's site for Derby is: **http://db.apache.org/derby/**

Apache PDFBox: An open source library for processing PDF documents. With PDFBox's help, one can create PDF documents, read existing documents, and extract information from them. Powergy is using PDFBox in order to obtain the number of pages for a given PDF document. Apache's site for PDFBox can be reached at: **http://incubator.apache.org/pdfbox/**

Apache POI: According to Apache's site for POI, the project "consists of APIs for manipulating various file formats based upon Microsoft's OLE 2 Compound Document format using pure Java". Powergy uses POI in order to process Word (DOC) documents and retrieve their number of pages. POI's source code, code samples and other documentation can be found at: **http://poi.apache.org/**

Deployment

Before one deploys Powergy successfully, there are two property files that must be updated depending on the target machine. These files are *printers.config* and *printers_energy.config*. Both files are located under the directory *capstone/powergy/src/edu/rit/powergy/main*. Here's an explanation on the syntax of these files:

printers.config: This file contains the name of the printers available to the system and their printing speed (i.e. pages per minute—PPM). The format is [printer_name]:[speed]. Here's an example: HP Color LaserJet 4700:31. This file is necessary since PPM is not a property you can obtain via the Java Printing API. Also, the name of the printer in the file must be equal to the name of the printer in the system. You can check your printers' unique names in your system's preferences. If there's no match, Powergy will assume the PPM is not defined for a given printer and assign 0 to this value. Realistically, printing speed relies on page content. For instance, complex graphics and images print slower than text. There's also the time it takes for a printer to go from standby to operational mode. These times all differ depending on the printer and its manufacturer. Since the Java Print API was not returning this attribute for any of the printers Powergy was tested on, this file was included to allow Powergy to deal with time estimates too.

printers_energy.config: The *printers_energy.config* file contains the initial values for the energy model associated to each printer. The format is [printer_name]:[class_name]:[value]:[composite_name]. Here's an example: HP Color LaserJet 4700:DeinkingEnergy:0.45:Page. Similarly to the *printers.config* file, the name of the printer in the file must be equal to the name of the printer in the system. If Powergy cannot parse the file, or there's a typo in the name of the class or the name of a printer, a value of 0 would be assigned to the metric at hand.

In order to make the compilation process easier, an Ant build script is provided (i.e. *capstone/powergy/build.xml*). Ant is nothing but a Java based 'make' tool which compiles the Java programs, links it with libraries, and creates executable files. Ant can be downloaded from **http://ant.apache.org/**. Once Ant has been set up correctly, typing ant in a terminal window under *capstone/powergy* would build the application. After the build process is complete, one can run Powergy by typing the following in a terminal window while at *capstone/powergy*:

```
java –jar powergy.jar
```

Testing

Several parts of the application can be tested separately. For instance, the reader could test Derby's ability to write to the hard disk with *DBTest.java,* or check if a particular printer is accepting print jobs with *PrinterUtilTest.java*.

Additionally, all the tests provided were constructed following a black-box approach. This essentially means every test case is taking an external perspective of the given test object when deriving test cases. Because of this disconnect between the test object and the implementation, black-box testing can pinpoint unimplemented aspects of the specification. Nevertheless, a negative facet of this approach is that, usually, one cannot guarantee that every existent path is tested and accounted for.

Javadocs

Javadocs are HTML documents generated automatically from Java source code by Sun Microsystems's javadoc tool. The application's javadocs can be found in *capstone/powergy/dist/javadoc*. Opening the *index.html* file in any web browser will display the documentation for all the packages and classes.

Summary

Powergy is the first step towards the full realization of a print life-cycle decision tool. As developed, Powergy focuses on energy consumption. But the processes, architecture and overall idea can be applied to expand the application and incorporate other estimates as well. For instance, in order to estimate water consumption or solid waste produced, one would need a water consumption model and a solid waste model. These models would have all the classes and the logic required to interact with any devices necessary (i.e. *Printer* in this case) for estimating consumption based on the attributes obtained from the printer, the document and the print job. The models are required because they're the rules that dictate how to compute an estimate, as well as what attributes are required from each component to perform the calculations. Aforementioned models would then communicate the estimate to the *Adviser*. Finally, the *Adviser* would then present the estimate to the user.

The tool was developed with future development in mind so that other developers can add more models and metrics easily. There are, however, aspects that could still be enhanced. Because of this situation, a look ahead is required. From a usability standpoint, the application would benefit from an enhanced user interface. If end users were able to better visualize the inner structure of the energy model and what each element entails, the estimate provided would carry more weight. More specifically, the energy information would help the end user better understand the impact of their print decisions, which may impact their behavior through more sustainable printing decisions.

Also, if Powergy were able to find a printer's capabilities and match energy metrics during runtime (i.e. plug and play), the setting up of the energy model via configuration files wouldn't be necessary. This would make deploying the application much easier and straightforward. The Java Print API may not be the best candidate since few, if any, of a printer's actual capabilities are found using Sun's interface.

On the other hand, if one were to think about a three-stage continuum for print life-cycle decision tools with *informing*, *advising* and *automatically deciding* as its constituents, Powergy would fall under the first category. Presently, the application is able to notify end users about the energy estimate of their decisions. Powergy is essentially reacting to end users' selections, and recalculating the estimated energy consumption after interactions with the UI have taken place. Yet, the application is actually not *suggesting* better courses of action based on those selections (with the exception of a small disclaimer at the bottom of the main form, which highlights the most frugal of the printers). *Advising* or *suggesting* more efficient printer and print job attributes to the end user is the second stage. A move from solely *informing* to *advising* would be ideal. The third and last stage—*automatically deciding*—is the idyllic goal. This sapient-like system would have to entertain many different possibilities and react according to end users' intent. The second or third stage would be the next logical, evolutionary step for a print energy life-cycle decision tool such as Powergy. The energy model was designed and

implemented to make this progression straightforward without having to modify the energy model.

Allowing end users to add/modify energy metrics and their computations would entail further development. Presently, Powergy is handling the definition and creation of new metrics via code. If this functionality will be available for end users, a new user interface would have to be created. This new UI would essentially replicate the steps a developer would take, outlined in the section *Extending and Modifying Powergy*, in an intuitive and easy to follow manner.

The ability to have multiple users is desirable as well. Powergy only supports one user at a time. If one wanted to add support for multiple users, a user administration module of some sort would have to be built into Powergy. Also, the current UI would have to change to accommodate the fact that various users could be using the application at the same time (e.g. ability for users to log in/out, capabilities so that users can update their profiles, etc.).

Lastly, Powergy would be a much more compelling prospect if it allowed for the use of different metrics other than just energy. For instance, computing CO_2 emissions, estimating water consumption, or approximating solid waste generated from the act of printing a document would move Powergy closer to the ideal print life-cycle decision tool. Additional metrics could be designed using the same composite and other design structures as the energy metrics.

References

Carli, D. (2007). Great print sustained. *Graphic Arts Monthly, 79(8),* 22.

Chadwick, P. (2008). Visions of a fruitful future. *Printweek,* 35.

Cross, L. (2008). Gathering on green print trends. *Graphic Arts Monthly, 80(5),* 51.

Curtin, D. (2008). Earth-friendly print. *Paper, Film and Foil Converter, 82(9),* 42.

Jones, G. (2008). Is print green? We'll soon find out. *Graphic Arts Monthly, 80(3),* 12.

Kemper, T. (2008). In pursuit of sustainability. *Print Professional, 46(7),* 12.

Lasater, C. G. (2007). *Design patterns.* Plano, TX: Wordware Publishing, Inc.

Weil, N. (2008). Fit to print. *CIO, 21(11).*

Appendix A: Existing Technology and Software

These are a few tools and products related to energy consumption found in the public domain:

Google's PowerMeter: Google is putting forward a new way of bringing energy consumption awareness to the public. PowerMeter would allow people to find out how much energy they're using, as well as to identify potential energy savers along the way. The project is a combination of both hardware and software tools. PowerMeter is still just a beta. More information can be found at: **http://www.google.org/powermeter/**

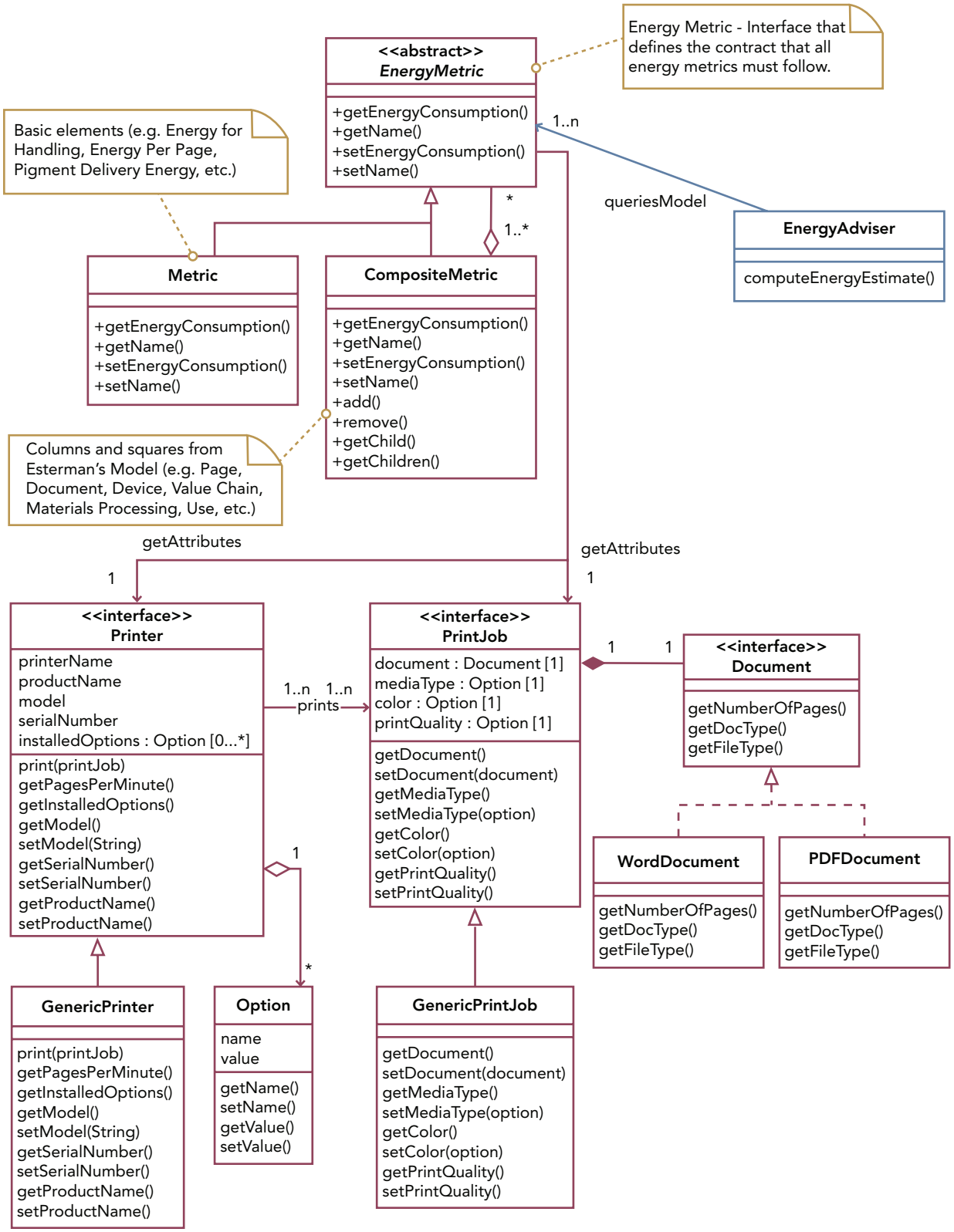
Energy Calculators and Software: The US Department of Energy has an extensive list of online calculators and tools about energy use. The list includes calculators for buildings, homes and vehicles. A good example is the Energy Efficient Rehab Advisor, which, according to the U.S. Department of Housing and Urban Development web site, "provides guidelines and savings information for incorporating energy efficiency into renovation projects for single and multifamily housing." The entire catalog can be found at: **http://www1.eere.energy.gov/calculators/**

IBM's Software Energy & Environment Self-Assessment: IBM has a comprehensive set of tools and services that run the gamut from ideal paper usage (according to an organization's type) to energy consumption. More information can be found at: **http://www.ibm.com/software/green**

Xerox has a sustainability calculator that helps print users to measure their greenhouse gas emissions, energy consumption, and solid waste generation. This calculator takes a measurement of the production and operation emissions of a laser or inkjet printer based on an average number of prints made per month and whether it is Energy Star approved. More information can be found at: **http://www.consulting.xerox.com/flash/thoughtleaders/suscalc/xeroxCalc.html**

Hewlett Packard also has a sustainability calculator that looks at the emissions produced by different printer models currently on the market. The calculator also shows the effects of consolidating devices or using energy saving technologies. More information can be found at: **http://www.hp.com/large/ipg/ecological-printing-solutions/carbon-footprint-calc.html**

Appendix B: Class Diagram



Since having all the concrete classes together with the interfaces would make the class diagram exceedingly big, only the interfaces are presented above. Here's the complete list, including the interfaces and the concrete classes that implement them:

EnergyMetric [interface]
CompositeMetric [concrete]
DeinkingEnergy [concrete]
EnergyForHandling [concrete]
EnergyForMaterialTransfer [concrete]
EnergyForPulpProcessing [concrete]
EnergyForTonerCartridge [concrete]
EnergyForTonerChemical [concrete]
EnergyPerPage [concrete]
EnergyRequiredForCultivation [concrete]
EnergyRequiredForHarvesting [concrete]
JobReclamationEnergy [concrete]
PigmentDeliveryEnergy [concrete]
PingmentFixingEnergy [concrete]
PostProcessingEnergy [concrete]
RecyclingEnergyForPaper [concrete]
StandbyEnergy [concrete]

Document [interface]
WordDocument [concrete]
PDFDocument [concrete]

Printer [interface]
GenericPrinter [concrete]

PrintJob [interface]
GenericPrintJob [concrete]

Option [interface]
GenericOption [concrete]

Appendix C: Java Printing API Printout

PRINTER: HP Color LaserJet 4700
copies = 1
finishings = none
Values for finishings: none
job-sheets = none
Values for job-sheets: none
Values for job-sheets: standard
media = na-letter
Values for media: na-letter
Values for media: na-legal
Values for media: executive
Values for media: invoice
Values for media: folio
Values for media: iso-a4
Values for media: iso-a5
Values for media: jis-b5
Values for media: folio
Values for media: oufuko-postcard
Values for media: ROC 16K
Values for media: na-number-10-envelope
Values for media: monarch-envelope
Values for media: iso-c5
Values for media: iso-designated-long
Values for media: iso-b5
Values for media: Custom
Values for media: Tray 1
Values for media: Tray 2
Values for media: Tray 3
Values for media: Tray 4
Values for media: Tray 5
Values for media: Tray 6
Values for media: Tray 1 (Manual)
number-up = 1
Values for number-up: 1
Values for number-up: 2
Values for number-up: 4
Values for number-up: 6
Values for number-up: 9
Values for number-up: 16
orientation-requested = portrait
Values for orientation-requested: portrait
Values for orientation-requested: landscape
Values for orientation-requested: reverse-landscape

Values for orientation-requested: reverse-portrait
page-ranges = 1-2147483647
Values for page-ranges: 1-2147483647
sides = one-sided
Values for sides: one-sided
Values for sides: two-sided-long-edge
Values for sides: two-sided-short-edge
media-printable-area = (4.233,4.276)->(211.624,275.167)mm
Values for media-printable-area: (4.233,4.276)->(211.624,275.167)mm
Values for media-printable-area: (4.233,4.276)->(211.624,351.367)mm
Values for media-printable-area: (4.233,4.276)->(179.874,262.467)mm
Values for media-printable-area: (4.233,4.233)->(135.467,211.624)mm
Values for media-printable-area: (4.233,4.233)->(211.582,325.529)mm
Values for media-printable-area: (4.233,4.276)->(205.698,292.693)mm
Values for media-printable-area: (4.233,4.276)->(143.722,205.74)mm
Values for media-printable-area: (4.233,4.276)->(177.758,252.561)mm
Values for media-printable-area: (4.233,4.276)->(211.582,325.882)mm
Values for media-printable-area: (4.233,4.276)->(143.679,195.665)mm
Values for media-printable-area: (4.233,4.276)->(192.532,268.732)mm
Values for media-printable-area: (4.233,4.276)->(100.457,237.067)mm
Values for media-printable-area: (4.233,4.276)->(94.107,186.267)mm
Values for media-printable-area: (4.233,4.276)->(157.692,224.705)mm
Values for media-printable-area: (4.233,4.276)->(105.706,215.731)mm
Values for media-printable-area: (4.233,4.276)->(171.662,245.703)mm
spool-data-destination= file:/Users/elvismontero/Desktop/capstone/powergy/out.ps
chromaticity = color
Value for chromaticity: color

Rochester Institute of Technology
College of Imaging Arts and Sciences
55 Lomb Memorial Drive
Rochester, NY 14623
Phone: (585) 475-2733
http://print.rit.edu