

5-26-2022

Metasoftware: Building Blocks for Legal Technology

Houman Shadab
New York Law School

Follow this and additional works at: <https://digitalcommons.law.seattleu.edu/sjteil>



Part of the [Business Organizations Law Commons](#), [Computer Engineering Commons](#), [Computer Law Commons](#), and the [Internet Law Commons](#)

Recommended Citation

Shadab, Houman (2022) "Metasoftware: Building Blocks for Legal Technology," *Seattle Journal of Technology, Environmental & Innovation Law*. Vol. 12: Iss. 2, Article 3.

Available at: <https://digitalcommons.law.seattleu.edu/sjteil/vol12/iss2/3>

This Article is brought to you for free and open access by the Student Publications and Programs at Seattle University School of Law Digital Commons. It has been accepted for inclusion in Seattle Journal of Technology, Environmental & Innovation Law by an authorized editor of Seattle University School of Law Digital Commons.

Metasoftware: Building Blocks for Legal Technology

Cover Page Footnote

Professor of Law and Director, Innovation Center for Law and Technology, New York Law School. Fellow, Stanford CodeX. B.A. 1998, University of California at Berkeley; J.D. 2002, University of Southern California.

Metasoftware: Building Blocks for Legal Technology

Houman Shadab*

I. INTRODUCTION

Recent years have seen a vast increase in the amount and types of software technology being employed by lawyers. The types and applications of technology include automated document drafting, contract workflow automation, and artificial intelligence empowered analytics and decision making applied to a wide variety of fields such as contract review and litigation strategy. This widespread adoption of software is due in large part to its increasing computational power and hence value to users.

This Article argues that further waves of software adoption will be driven not only by increasing computational power, but also by increasing the accessibility and enabling power of software development tools. Innovation in software is not only about computational processing power, but also about enabling a wider range of users to create their own powerful software without significant technical expertise. This Article is the first to identify the recent emergence of such “metasoftware.”

Metasoftware enables users to create the software of their choosing. It stands in sharp contrast to traditional, functional software that is intended for a particular purpose or a defined range of tasks. Functional software is the default type of software that is currently produced and includes word processing, email, social networking, enterprise resource management, online marketplaces, and video game software. Metasoftware, by contrast, is *not* functional. Metasoftware presents the user with a blank slate upon which to build functional software.

The quintessential manifestation of metasoftware is the recently emergent phenomenon known as the “no code revolution” that frees users from being required to manually write code to create and customize software. These “visual software development platforms” automate the code writing process using drag and drop, menus, flowcharts, and other

* Professor of Law and Director, Innovation Center for Law and Technology, New York Law School. Fellow, Stanford CodeX. B.A. 1998, University of California at Berkeley; J.D. 2002, University of Southern California.

visual elements. Visual software development platforms are metasoftware because they present users with precisely the type of blank slate upon which to create functional software.¹

As technology continues to proliferate throughout every aspect and role of legal practice and professional services more broadly, attorneys will increasingly find themselves not just using technology for its existing functional purposes, but also creating their own software and customizing existing software using metasoftware. Pre-packaged, off-the-shelf functional software will increasingly be too limited and expensive for lawyers to deliver to highest quality technology-enabled legal services to clients. Clients will demand software that is highly customized for their particular needs and can rapidly change along with their needs. Accordingly, lawyers will not only increasingly use technology as a tool of practice, but they will also increasingly create and customize their own software applications and customizations.

Section II of this Article develops a novel theory of metasoftware as distinguished from traditional, functional software. I argue that software is metasoftware to that extent that (1) it enables users to build user interface elements, workflow logic, and perform database operations; (2) provides connectivity with external data and software systems; and (3) can be stored and run independently from the platform that is used to build the software.

In its purest form, metasoftware enables its users to build any functional software (given the existing state of technology); integrate with all open software platforms; and be hosted and run in the environment of the user's choosing without being bound to a particular vendor or other proprietary software platform. My identification of metasoftware contributes to the academic literature on information systems and technology. Metasoftware is a hitherto unrecognized category of software for analysis in terms of several foundational lines of information technology research including user acceptance and usage, diffusion within an organization, and impact on organizational innovation and success (e.g., business performance).

Section III then analyzes several existing visual software-development platforms to determine the extent to which the platforms qualify as metasoftware. The first category are platforms for building functional software that are closely tied to major producers of cloud-based software platforms such as Microsoft's Power Platform and Google's AppSheet. Power Platform and AppSheet enable users to build a very wide range of functional software products but are limited in their ability to integrate with other systems and cannot operate outside of their host platforms. The second category are standalone proprietary "no code" software builder platforms typified by Bubble. These standalone platforms enable users to build an extremely broad range of functional software applications and integrate with other systems, but generally do not enable users to run the functional software they build with such platforms in other environments. Platforms in the third category are closest to the ideal of

¹ Code itself is not metasoftware, but rather used to create metasoftware.

metasoftware: open-source visual development platforms where the technology that is used to build the metasoftware platform is itself completely transparent and nonproprietary. The benefits of open-source in this context are to enable full user control over where the functional software they build is stored and operated. However, open-source platforms may offer users limited ability to produce functional software relative to proprietary platforms motivated by profit. Accordingly, Section III identifies important tradeoffs in terms of the three metasoftware characteristics between the three types of metasoftware platforms. These tradeoffs in turn impact how each type of platform should be used as building blocks for “legaltech” software, which are discussed in Section IV.

Section IV explains how metasoftware platforms can be used to build functional legal technology. This section focuses on four major categories of legaltech as illustrative of the potential for metasoftware to build functional software: legal research, legal matter management, contract automation, and a variety of applications of artificial intelligence. As recently argued in the article *Augmented Lawyering*, which was presented at the Yale Law School Center for the Study of Corporate Law, AI “will augment the capabilities of human lawyers who use AI-enabled services as inputs to their work and generate new roles for legal experts in producing these AI-enabled services.”² This Article accordingly also contributes to the literature on how technology is transforming legal practice to argue for how such technology-enabled legal services can and should be created by lawyers and other legal professionals.

Section V concludes by way of summary and recommendations regarding what type of metasoftware is best suited for building certain functional legaltech applications.

II. A THEORY OF METASOFTWARE

Metasoftware enables users to create the software of their choosing and stands in sharp contrast to traditional, functional software that is intended for a particular purpose or a defined range of tasks. Functional software is the default software that is currently produced and includes word processing, email, social networking, enterprise resource management, online marketplaces, and video game software. Metasoftware, by contrast, is *not* functional. Metasoftware presents the user with a blank slate upon which to build functional software.

In its purest form, metasoftware enables its users to build any functional software (given the existing state of technology), integrate with all open software platforms, and be hosted and run in the environment of the user’s choosing without being bound to a particular vendor or other proprietary software platform.

This Section details the characteristics of metasoftware and discusses how the emergence of metasoftware has important implications for information technology research.

² John Armour, Parnham Richard, and Mari Sako, *Augmented Lawyering* (European Corporate Governance Institute, Law Working Paper No. 558, 2020).

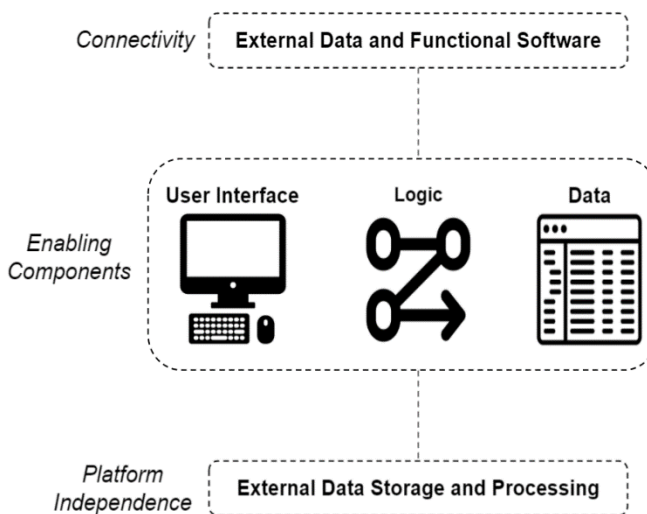
A. Metasoftware and Code

1. Structure of Metasoftware

Software should be viewed as metasoftware to the extent that it has enabling, connectivity, and platform independence characteristics. The enabling aspect of metasoftware means that it provides users with substantive ability to build user interface elements, workflow logic that executes computer programming tasks, and engage in database storage operations.³ The connectivity aspect of metasoftware exists to the extent that the software integrates with external data and software systems. Finally, the platform independence aspect of metasoftware exists to the extent that functional software and its associated data and code is able to be stored and run (i.e., processed) independently of the metasoftware platform that is used to build the software.

The enabling, connectivity, and independence characteristics of metasoftware are illustrated in Figure 1.

Figure 1: Metasoftware Characteristics and Structure



The conceptual and functional dividing lines between these characteristics are not always easy to distinguish. For example, the ability of a metasoftware platform to connect to an external database effectively enables that data to become part of (or internal to) the data that is used by the functional software being built on its platform. In this sense, connectivity enables data storage and operations. Similarly, platform independence may also enable the processing of unique workflows (such as those enabled with blockchain technology) that are only capable of

³ See Ariel Ortiz Ramirez, Three-Tier Architecture, LINUX JOURNAL (July 1, 2000), <https://www.linuxjournal.com/article/3508> [<https://perma.cc/9ES6-XLPW>] (this is consistent with the widely used three-tier software architecture.)

being executed on platforms that are distinct from the metasoftware platform. In this sense, platform independence enables substantive workflows.

2. *Visual Software Development and the Role of Code*

Generally, a “software program” is a set of rules or step-by-step instructions executed by a computer to perform a task. These tasks produce an output or have some kind of action in response to processing an input.⁴

Given that computer hardware can only “understand” binary logic expressed in the form of 1s and 0s, all programming rules must ultimately be processed in the form of machine code, “a computer programming language consisting of binary or hexadecimal instructions which a computer can respond to directly.”⁵ Because of the inherent difficulty in programming a computer with machine code, numerous programming languages and paradigms have been developed that abstract away from 1s and 0s in order to make programming with code more accessible. When processed, however, all higher-level computer languages must ultimately compile or otherwise translate down to machine code to be executed. As explained by Michael Schmidt,

Machine language is the language understood by a computer. It is very difficult to understand, but it is the only thing that the computer can work with. All programs and programming languages eventually generate or run programs in machine language.⁶

Different programming languages differ in many ways, including with respect to their level of abstraction from machine code, their particular syntax, and how they conceptualize various rules and operations. These rules and operations include mathematical calculations, defining variables, structuring data, undertaking repetitive procedures (e.g., algorithms), and the order in which computational tasks should take place.⁷ Unsurprisingly, different languages are better suited for certain

⁴ See e.g., ASHOK ARORA, *COMPUTER FUNDAMENTALS AND APPLICATIONS* 74 (Vikas Publishing House ed. 2015). (this does not mean that computational tasks are inherently deterministic such that the same input(s) will always produce the same output in a given program. Programming languages can be ambiguous). See James Grimmlemann, *All Smart Contracts Are Ambiguous* 2, *LAW JOURNAL OF INNOVATION* 1, 11-14 (2019). (a set of programming instructions that perform a specific task is a procedure).

⁵ *Machine Code*, *ENCYCLOPEDIA.COM* (May 23, 2018), [https://www.encyclopedia.com/science-and-technology/computers-and-electrical-engineering/computers-and-computing/machine-code#:~:text=ma%C2%B7chine%20code%20\(also%20machine,computer%20can%20respond%20to%20directly](https://www.encyclopedia.com/science-and-technology/computers-and-electrical-engineering/computers-and-computing/machine-code#:~:text=ma%C2%B7chine%20code%20(also%20machine,computer%20can%20respond%20to%20directly) [<https://perma.cc/ME7Y-W8JA>].

⁶ *Machine Language*, *SCIENCE DIRECT*, <https://www.sciencedirect.com/topics/engineering/machine-language> [<https://perma.cc/EUX7-3Q3C>] (last visited March 22, 2022, 12:00 pm), citing MICHAEL L. SCHMIT, *PENTIUM™ PROCESSOR* (1st ed. 1994).

⁷ HAROLD ABELSON ET AL., *STRUCTURE AND INTERPRETATION OF COMPUTER PROGRAMS* 6-15 (2nd ed. 1996).

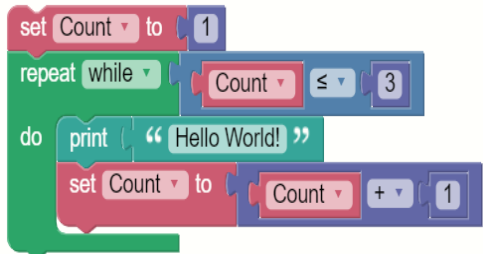
tasks, such as JavaScript for web applications and Python for machine learning.⁸

A recent development has been the proliferation of powerful, customizable, visual programming platforms that do not require a programmer to write any code. Visual programming tools operate at the highest level of abstraction above traditional code-based programming languages. Typically, a visual programming platform auto-generates traditional code in response to users establishing aspects of a software such as workflow logic and database interactions. This process is described by the no code platform Ycode:

While you drag and drop elements into the canvas and style them to perfection, Ycode silently writes clean code based on preferred open-source frameworks like Laravel and TailwindCSS.⁹

A very simple example of visual software development compared to traditional code-based development is shown in Figure 2 using the Google Blockly platform. The visual programming component for displaying “Hello World!” until the “OK” button is pressed three times is displayed next to its equivalent in the JavaScript programming.¹⁰

Figure 2: Visual Programming Versus Code Programming

Visual Programming	JavaScript Code
 <p>The image shows a Blockly script. It starts with a 'set Count to 1' block. This is followed by a 'repeat while' loop. The loop's condition is 'Count ≤ 3'. Inside the loop, there is a 'do' block containing two steps: 'print "Hello World!"' and 'set Count to Count + 1'.</p>	<pre>var Count; Count = 1; while (Count <= 3) { window.alert('Hello World!'); Count = Count + 1; }</pre>

With visual development, users do not need to be concerned with the syntax of any particular programming languages. Visual software components are both symbolic and functional, and therefore promote linguistic meaning with action-functionality. As noted by the no code platform Unqork, “[b]y fully abstracting code into a completely visual interface, users can rely on [graphical] components to build

⁸ See JOHN M. ZELLE, PYTHON PROGRAMMING: AN INTRODUCTION TO COMPUTER SCIENCE 8 (2004). (different languages may fall within broader programming paradigms such as object-oriented programming or functional programming. Ultimately, all human-readable programming languages must be converted (“compiled”) into binary notation so that its rules and data can be processed by the computer’s circuits (which can only process 0s and 1s)).

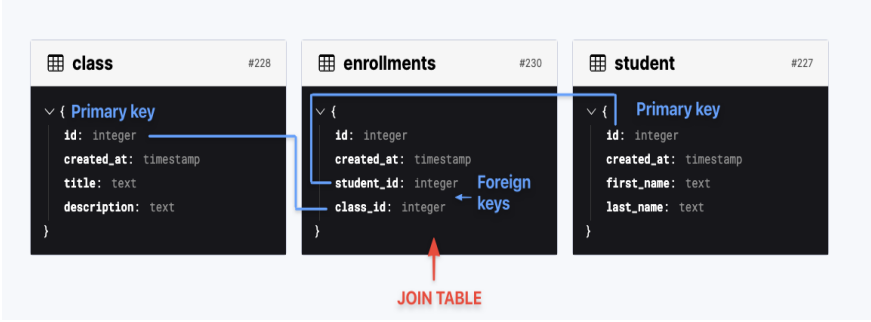
⁹ YCODE, <https://www.ycode.com> [<https://perma.cc/M72R-RYRZ>] (last visited March 12, 2022).

¹⁰ TRY BLOCKLY, <https://developers.google.com/blockly> [<https://perma.cc/V2MS-7B83>] (last visited March 12, 2022).

applications . . . No-code allows us to work in natural human languages.”¹¹ With visual development, “ideas are immediately turned into [software applications] that work the way they should intuitively, allowing users to focus on ideas over the syntax.”¹² This tight coupling between meaning and software interface enables the blank slate feature of metasoftware to emerge; users can build in a truly outcome-oriented fashion focused solely on function and not non-functional aspects of software code.

In the context of databases, visual software development enables the use of Entity Relationship Diagrams (ERD) and other visual database tools that permit data processing “in ways that are intuitive to the human brain . . . ”¹³ An ERD defines entities and their attributes, and shows the relationships between them to illustrate the logical structure of databases.¹⁴ An example ERD for a database that organizes data about students and classes is shown in Figure 3.¹⁵

Figure 3: Entity Relationship Diagram Example



The following figure shows the relationship between the highest level of abstraction visual software development platforms and the lowest-level machine code, including the various standard types of programming languages in between based on the abstraction layer.

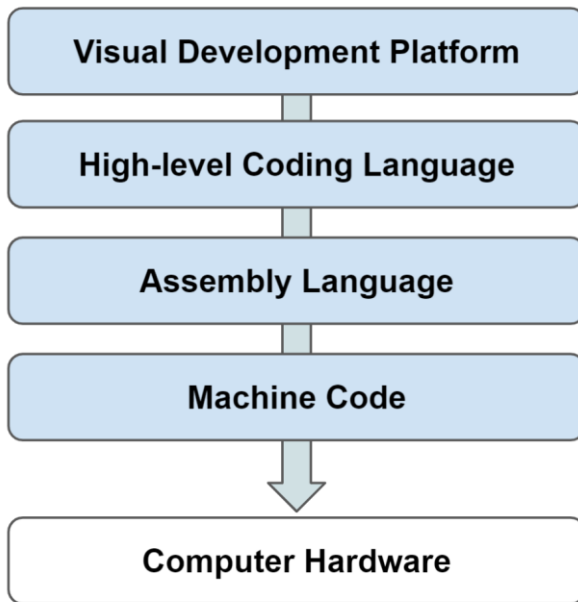
¹¹ *Welcome to the Era of No-Code in the Enterprise*, UNCORK RESOURCE CENTER, <https://www.unqork.com/resources/articles/welcome-to-the-era-of-no-code-in-the-enterprise> [https://perma.cc/9ZAK-U644]. See also, *Explaining No Code to Engineers*, Uncork Resource Center, (“No-code’s visual workflows create a common language that both business users and engineers can speak.”), <https://www.unqork.com/resources/blog-articles/explaining-no-code-to-engineers> [https://perma.cc/5Z4F-JQY7].

¹² *Humanizing Development with No-Code*, UNCORK RESOURCE CENTER, <https://www.unqork.com/resources/blog-articles/humanizing-development-with-no-code> [https://perma.cc/3SVL-7MTE].

¹³ *Id.*

¹⁴ *SmartDraw*, ENTITY RELATIONSHIP DIAGRAM, <https://www.smartdraw.com/entity-relationship-diagram> [https://perma.cc/QPN4-EGGP].

¹⁵ *The Database: Database Relationships*, XANO, <https://docs.xano.com/database/database-relationships> [https://perma.cc/GB2T-3DWY].

Figure 4: Programming Language Abstraction Hierarchy

Programming language abstraction has two important implications for metasoftware.

First, metasoftware must use visual software development as a primary tool of development. This is because visual software tools are required for metasoftware to have the emergent property of being a blank slate upon which to build functional software. Although functional software is traditionally built by writing code and without visual development tools, any particular environment for creating functional software by writing code is too restrictive in what it enables compared to a visual development platform.¹⁶

Second, the hierarchy of programming languages is also important in understanding the platform independence characteristic of metasoftware. The independence of metasoftware rests upon the ability of the functional software that is created on the metasoftware platform to be able to run on external platforms. Running on external platforms means that the code underlying the functional software is able to be compiled to and run on other platforms. Platform independence is also supported by the ability of the visual development tools of a particular metasoftware platform to be replicated and run on other metasoftware platforms distinct from the actual functional software that they are used to create.

¹⁶ This is because coding platform environments at most focus on one specific component of software, such as UI or a subject or workflow logic. In doing so, they cannot be conceptualized as a single platform entity that can be used to make software. Several strands of code must be combined together in order to create functional software whereas only one visual development platform is required. Indeed, code by itself is not software, but rather what is required to make functional software. Because visual development is at such a high level of abstraction, it crosses over numerous subdomains of code to produce functional software.

B. *Characteristics of Metasoftware*

1. *Enabling Components*

a. *User Interface Elements*

A software application's user interface is the visual means by which users view and interact with an application's data and programmable logic. UI elements include all of the textual and graphical elements that are able to communicate scholarship:

- text fields,
- buttons that initiate actions such as navigating the user to another page, or saving input data in a database,
- input fields,
- icon, images, video, other graphical, and design elements,
- elements that group or otherwise organize other elements,
- navigation elements such as sliders,
- standardized elements for uploading files and inputting date/time,
- graphs, charts, timelines, and other data-oriented elements.

UI elements have properties. These properties include characteristics such as size, page position, color, and how they may behave when certain conditions are true or in response to user actions. User interaction with UI elements, such as by clicking on them or entering text into a box can also initiate programming tasks. These actions are governed by the logic programmed into the application and may or may not interact with the application's database. UI elements may also have data attached to them (known as a "state") that does not reside in any database.

Data visualization tools are types of UI elements that provide users with coherent and meaningful presentations of large volumes of data and complex findings to show trends, relationships, and highlight significance. As shown in Figure 5, these data visualization tools include a wide variety of visual formats for data ranging from standard pie charts and scatter plots diagrams to bubble charts and network diagrams.¹⁷

¹⁷See Severino Ribeca, THE DATA VISUALIZATION CATALOGUE, <https://datavizcatalogue.com> [<https://perma.cc/ARL7-8CSX>]. See also *Digital Humanities: Visualizations*, NEW YORK UNIVERSITY, <https://guides.nyu.edu/digital-humanities/tools-and-software/visualization> [<https://perma.cc/5M2F-7LP7>].

Figure 5: Types of Data Visualization



b. Workflow Logic

The heart of any software application is its programmable logic that enables the application to undertake tasks. Programmability creates interactivity between a user and a software application because the application is instructed to undertake a task in response to user inputs or changes in external data.

A programmed sequence of events that begins with a triggering event that causes a software output is a *workflow*. Programmable workflows implement the logical operators and other fundamental computer science concepts. Triggering events include (1) user interface triggers (e.g., pressing a button, entering text in a text box element, a change in another element) and (2) changes to or evaluations about data in an internal or external database (e.g., two days have elapsed since an email was sent. A new document was uploaded to a database). Actions that take place *in response to* triggers include (1) changes being made to a user interface element (e.g., new text based is displayed, color changes to a graphic, the appearance of an image); (2) changes being made to internal or external data; and (3) the performance of an operation such as performing an operation on data. A second type of workflow structure is maintaining a property or (continuing to) performing an action *while* or *so long as* a condition remains true. For example, users may not be permitted to access certain data so long as they have not passed a security test.

Rules that govern workflows can also be made subject to exceptions so that the workflow only operates (or does *not* operate) when certain triggers take place or conditions are true. Workflows also enable significant automation when multiple actions take place in response to a fewer number of triggers. Ultimately, an application’s underlying programming logic governs the connection between a trigger and resulting action.

c. Database Operations

Creating functional software involves organizing information relating to the subject of the work in a digital format—as data. This is because software operations that involve the use of information necessarily involve the storage, processing, and display of data. Not surprisingly, “computer scientists have developed a large body of concepts and techniques for managing data.”¹⁸ Accordingly, a foundational aspect of software development is properly establishing how data is structured, stored, processed, and displayed by a software application. Modern databases facilitate everyday activities such as online transaction processing and data analytics.¹⁹ A primary purpose of a database system is to provide users with an abstract view of data and hide details about how data is stored and maintained.²⁰ Database systems are used to manage collections of data that are relatively large, valuable, and able to be accessed by multiple users or applications.²¹

In building functional software, metasoftware users will most likely use relational databases when storing data within an application and nonrelational (or semi-structured) databases when accessing data available on external websites or platforms. A relational data model organizes data into tables and is characterized by spreadsheets.²² In a relational model, the same type of data must have the same attribute. For example, all rows have the same columns. By contrast, in a semi-structured data model, “individual data items of the same type may have different sets of attributes.”²³ Semi-structured data is more flexible than relational data, enables the organization of a wide variety of data types, and facilitates online data transmission because it is a standard data format for transmitting data over the Internet. The data object shown in Figure 6 shows an example of semi-structured data in the popular online data format known as JavaScript Object Notation:²⁴

¹⁸ABRAHAM SILBERSCHATZ ET AL., *DATABASE SYSTEM CONCEPTS* 1 (7th ed., 2020).

¹⁹ *Id* at 4 (“the processing of data to draw conclusions, and infer rules or decision procedures, which are then used to drive business decisions”).

²⁰ *Id* at 29 (“A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained”).

²¹ *Id* at 2.

²² *Id* at 8.

²³ *Id*; See also Hastie, Chris, *Zero to Snowflake: An Introduction to Semi-Structured JSON Data Formats*, INTERWORKS (Jan. 21, 2020), <https://interworks.com/blog/chastie/2020/01/21/zero-to-snowflake-an-introduction-to-semi-structured-json-data-formats/> [https://perma.cc/C5VJ-KDBQ].

²⁴ *How to Fetch and Display JSON Data in HTML Using JavaScript*, HOW TO CREATE APPS (Mar. 22, 2022), <https://howtocreateapps.com/fetch-and-display-json-html-javascript/> [https://perma.cc/A24G-SW6U].

Figure 6: Example of Semi-Structured Data

```
{
  "id": "1",
  "firstName": "John",
  "lastName": "Doe"
},
{
  "id": "2",
  "firstName": "Mary",
  "lastName": "Peterson"
},
{
  "id": "3",
  "firstName": "George",
  "lastName": "Hansen"
}
```

The most common database operations enable users to:

- retrieve information stored in a database.
- insert new information into a database.
- delete information from a database.
- modify information stored in a database.²⁵

An important consideration is what level of access different types of users have to a database, or what functionality they are permitted to perform based on role or pricing.²⁶

Data is a foundational aspect of functioning software. By interacting with UI elements, users can view, add, edit, delete, and have calculations performed on an application's data. In addition, workflows can process, change, or otherwise involve data within an application's database. By using an API to connect to data that resides outside of an application, users can likewise create workflows that interact with data that is external to the application's database. Connecting to external APIs also enables functionality from third party applications to be integrated into an application's workflows, such as Google Maps, payments, and search functionality.

To create a database, one must first create a conceptual schema of how real-world objects or information should be categorized, relate to one another, and be described with data-oriented properties. *Data categorization* involves determining the appropriate level of abstraction to create a database. *Data relation* involves determining how to connect two or more databases together due to sharing common data points, if at all. *Data description* involves how to identify the properties of data objects with particular data points. This often entails deciding what column headings should be used in a standard database.

In creating an application about Shakespeare's work, for example, the design of the database will revolve around decisions about how to

²⁵ Silberschatz et al., *Supra* note 19, at 8. (these four operations parallel what is commonly known as CRUD database operations: "create, read, update and delete.") See also *Id.* at 419.

²⁶ Silberschatz, *supra* note 19, at 24-25.

categorize his work (e.g., by type or by theme), how his work may relate to each other so that common properties can be identified (e.g., an archetypal character that appears in many of his works), and how to describe any particular work with properties such as length, character motivations, and date published.

2. Connectivity

As a blank slate upon which to create functional software, an important aspect of metasoftware is its ability to connect to external databases and other software systems. Generally, modern software has high degrees of connectivity due to the open data movement and the application programming interface (API) revolution. Open data and API-enabled automation workflows enable users to connect and integrate an extremely wide array of external platforms, software-driven functionality and services, and real-time data sources.

The open data movement refers to the increasingly widespread availability of constantly updated, high-quality data about nearly every aspect of human society, especially from governmental and academic communities. The accessibility of public data was greatly enhanced in January 2020 when Google's Public Dataset search became publicly available, making millions of data sets searchable.²⁷ In parallel to the increasing availability of public data is the increasing accessibility of tools that enable users to employ the methods of data science, data visualization, and machine learning without code or formal training in statistics.²⁸ These tools can be used alongside or directly integrated within the user interface of visual application development platforms.²⁹

Machine learning and data science tools enable analysis of large data sets to find patterns and estimate on future outcomes based on data that reveals prior associations. For example, according to the no code data science tool Obviously AI,

²⁷ Natasha Noy, *Discovering Millions of Datasets on the Web*, (Jan. 23, 2020), <https://www.blog.google/products/search/discovering-millions-datasets-web/> [https://perma.cc/7EAP-TW3V].

²⁸ See, e.g., Jack Riewe, *How No-Code Machine Learning Algorithms Work*, OBVIOUSLY AI, INC. (May 14, 2020), <https://www.obviously.ai/post/how-no-code-machine-learning-algorithms-work> [https://perma.cc/B4AV-DPNY]; *Enrich Your Stories with Charts, Maps, and Tables*, DATAWRAPPER, <https://www.datawrapper.de/why-datawrapper/> [https://perma.cc/HBS4-8Z8Y] (last visited Mar. 23, 2022); *About Ludwig*, LUDWIG, <https://ludwig-ai.github.io/ludwig-docs/> [https://perma.cc/BW4S-7N67] (last visited Mar. 23, 2022) ("Ludwig is a toolbox that allows to train and test deep learning models without the need to write code."); *About Lobe*, LOBE, <https://lobe.ai/about> [https://perma.cc/V3VX-ESB8] (last visited Mar. 23, 2022); *The Wolfram Approach to Machine Learning*, WOLFRAM, <https://www.wolfram.com/featureset/machine-learning/> [https://perma.cc/W9SL-KL3E] (last visited Mar. 23, 2022) ("making state-of-the-art machine learning in a full range of applications accessible even to non-experts."); *RAWGraphs* – Introduction, YouTube (Feb. 22, 2017), <https://www.youtube.com/watch?v=2TtYlty-M5g> [https://perma.cc/J2GQ-AAEM] ("RAWGraphs is an open source data visualization tool built with the goal of making visualization of complex data easy for everyone."); *AutoML*, GOOGLE CLOUD <https://cloud.google.com/automl/> [https://perma.cc/VKR5-X7RH] (last visited Mar. 23, 2022). See also generally Rohit Chatterjee, *Top Ten Tools for No-Code AI & ML*, ANALYTICS INDIA MAGAZINE (Feb. 11, 2020), <https://analyticsindiamag.com/top-10-tools-for-no-code-ai-ml/> [https://perma.cc/D39E-K3DS].

²⁹ See, e.g., Allen Yang, *No-Code Deep Learning with Bubble & Peltarion*, BUBBLE GROUP, INC. (Jul. 16, 2020), <https://bubble.io/blog/bubble-peltarion-machine-learning-ai/> [https://perma.cc/TV3U-YVTJ].

a large insurer developed a report for predicting the likelihood that a workers' compensation claim would lead to litigation. Claims with probability of litigation were referred to senior staff for early and attractive settlement offers. This saved the company 8% of the litigations they would face otherwise and \$3 million annually.³⁰

The API revolution consists of the increasingly widespread practice of software and data being made available to be easily integrated inside (or connected to) with other applications without having to rebuild any of the functionality of the external software. A prominent example is Google making Google Maps available via API so that websites and other apps can use Google Maps in a way that is relevant to their business or other purpose without having to build their own global GPS-based map application. The Programmable Web directory lists of over 22,000 public APIs.³¹

Supporting the connection of software applications via API are visual connector platforms such as Zapier and Integromat. The benefit of connector apps is offering pre-made interconnections (often called "recipes") between different software systems. These pre-made connectors remove the need to read an application's API documentation to incorporate the functionality of an external data set or software system within another application. No code platforms also offer plug-ins that often make adding other applications' functionality even easier than using a connector platform. Connector tools like Zapier and other API-enabled automation workflows significantly enhance the connectivity of modern software to an extremely wide and ever-growing array of platforms and data sources.

3. *Platform Independence*

Software platform independence is a characteristic of software that is present to the extent the software (and any related programming language, data, or another digital phenomenon) is able to be stored and operated with any database type, operating system, or other computing environments.³² Software that is able to run on several systems or platforms is known as cross-platform software.³³

Platform independence is supported to the extent that the underlying code metasoftware is open source. Code is the open-source when the code is publicly available and available for third parties to view,

³⁰ Harish, *Insurance*, SCOOPML (Jun. 1, 2020), [https://www.scoopml.app/post/\[https://perma.cc/NC8E-BHGB\]](https://www.scoopml.app/post/[https://perma.cc/NC8E-BHGB]).

³¹ *ProgrammableWeb Research Center*, PROGRAMMABLEWEB, <https://www.programmableweb.com/api-research> [<https://perma.cc/DX5Z-3WHC>] (last visited Mar. 23, 2022).

³² See *Platform-Independent*, GARTNER, <https://www.gartner.com/en/information-technology/glossary/platform-independent> [<https://perma.cc/LQJ7-MZ3B>] (last visited Mar. 23, 2022).

³³ *Cross-Platform Software*, WIKIPEDIA, https://en.wikipedia.org/wiki/Cross-platform_software [<https://perma.cc/6BXP-M439>] (last visited Mar. 23, 2022).

edit, and use in their projects.³⁴ Open-source software supports platform independence because open-source software can be exported to systems outside of a proprietary platform or system. Proprietary software can also be exportable to external systems; in such a case the software underlying the platform will not qualify as open source but the code underlying the functional software application that a user builds with the metasoftware will be fully owned and editable. A benefit of the metasoftware platform itself being open source is that it may facilitate platform independence by making it easier for user to implement the code they export on alternative systems. This is because the user may be able to view and use their version of the metasoftware's code that interacts with the code associated with the functional software application.

Platform independence is supported on the deepest level of the infrastructure that runs software by decentralized computer architectures. Traditional infrastructures for running software and storing user data rely on centralized computing systems that are either owned by corporate users for their own purpose (such as a bank's own computer servers running the bank's software on its premises) or available over the internet via "cloud" networks that are operated by companies such as Amazon Web Services, Microsoft Azure, and Google Cloud. By contrast, decentralized computing systems are not operated by any specific entity. Decentralized computing operates through computing networks that operate according to protocol rules that prevent any entity from controlling the network operations or accessing user data by default. Decentralized computing supports platform independence because it gives users of metasoftware much more control over where and how the software they create operates, including independence from rules established by centralized computing systems. Decentralized computing is often referred to as Web3. It includes decentralized blockchain networks that keep data private, and transactions secure via cryptography and incentivize parties to provide computational power by earning cryptocurrency.³⁵ How particular Web3 projects further the characteristic of platform independence are discussed below.

C. *Metasoftware and Information Systems*

The emergence of metasoftware has several significant implications for the discipline of Information Systems (or Information Technology). Metasoftware is most relevant for the following categories of IS research: user acceptance and use of IT, the impact of IT on organizations, IS success, IT and business performance, adoption, and IT implementation. According to Heikki Topi, IT research attempts to answer foundational questions such as:

- Why do users choose to accept some technologies readily and others not at all?

³⁴ *What is Open Source?*, OPENSOURCE.COM, <https://opensource.com/resources/what-open-source> [https://perma.cc/5YB6-55NQ] (last visited Mar. 23, 2022).

³⁵ See generally Emre Tekisalp, *Understanding Web 3 — A User Controlled Internet*, COINBASE (Aug. 29, 2018), <https://blog.coinbase.com/understanding-web-3-a-user-controlled-internet-a39c21cf83f3> [https://perma.cc/43DM-CLQD].

- What are the mechanisms through which IT affects organizations and can be used to transform them?
- How do we define and measure the success of IS in organizations?
- Do IT-based solutions really have a measurable impact on business performance?³⁶

The largest area of research in IT is user acceptance and use of IT. The key proposition of this research is that “the probability of a user’s behavioral intention to use a specific IT is based on two primary factors: perceived usefulness and perceived ease of use, the former directly and the latter both directly and mediated by perceived usefulness.”³⁷ The more specific area of adoption and diffusion research focuses on attributes of IT that affect its adoption such as those identified by Everett M. Rogers: relative advantage, compatibility, complexity, observability, and trialability.³⁸ Metasoftware is relevant to the literature on IT user acceptance and usage.

Metasoftware is orders of magnitude more accessible than traditional IT systems due to metasoftware not requiring any expertise in coding or broader IT systems to be effectively used. This means that non-technical employees within an organization can use metasoftware to perform operations and undertake activities that would otherwise require IT professionals. These operations and activities include a wide variety of digital transformation tasks such as digitizing and creating a searchable repository for paper-based documents, automation of business processes such as communications and approvals and applying artificial intelligence and data analytics to decision-making. Metasoftware also scores very highly in all of the Rogers’ diffusion factors:

1. metasoftware is widely regarded as superior to traditional functional software (at least for a wide range of purposes);
2. metasoftware builds upon existing software frameworks and is compatible with them (e.g., custom code can be added to most metasoftware platforms);
3. metasoftware is less complex than coding-based systems;
4. metasoftware is highly triable because users can in just a few hours onboard to a system and begin to use it; and
5. metasoftware is highly visible to third parties because of the front end, user interface elements, and results that can be created with it.

IT research also focuses on the impact of IT on organizations (including its impact on innovation, emergent practices, and organization structure and change); its success (as measured by factors such as “system quality, information quality, use, user satisfaction, individual impact, and

³⁶ Heikki Topi, *Evolving Discipline of Information Systems in Computing Handbook*, Vol. 2: *Information Systems and Information Technology* 1, 12 (Heikki Topi & Allen Tucker eds., 3d ed. 2014).

³⁷ *Id.* at 3.

³⁸ EVERETT M. ROGERS, *DIFFUSION OF INNOVATIONS* 15-17 (5th ed. 2003).

organizational impact”); and how IT improves business performance (with studies viewing IT as a competitive advantage and contributing positively towards firm output and innovation).³⁹ In each of these areas, metasoftware presents a fundamental shift and contribution: it will help organizations much more rapidly innovate, it will improve IT success due to its wide-ranging accessibility, and it will improve business performance by dramatically lowering the costs of IT spending and vastly increasing the speed of IT.

III. SOFTWARE PLATFORMS AND INFRASTRUCTURE AS METASOFTWARE

This Section examines several existing software platforms and broader pieces of software infrastructure as examples of metasoftware or supporting infrastructure having the characteristics of enabling components, connectivity, and platform independence. These software platforms and infrastructure are:

- visual development platforms created by the largest technology companies in the world (“big tech”),
- standalone proprietary visual development software providers,
- standalone open-source visual development software providers, and
- emerging decentralized software infrastructure.

While none of the foregoing qualifies as pure metasoftware or infrastructure in the sense of possessing or enabling all of the characteristics of metasoftware as outlined in Section II, each possesses most of the characteristics to a sufficient degree to be classified as metasoftware. However, each of the foregoing possesses important tradeoffs relative to metasoftware characteristics. These tradeoffs for the platforms are summarized in the following Table:

³⁹ Heikki Topi, *Computing Handbook: Vol. 2: Information Systems and Information Technology* 1-6 (3rd ed. 2014).

Table 1: Metasoftware Characteristic Tradeoffs Among Different Platforms

	Cloud Provider Platforms	Standalone Proprietary Platforms	Open Source Platforms
Enabling Components	High	High	Medium to Low
Connectivity	High, especially for proprietary components	High, especially for wide ranging plugins	Low
Platform Independence	Low	Low	High

Importantly, these tradeoffs have important implications for building legaltech applications, which is examined Section IV.

A. Cloud Provider Visual Development Platforms

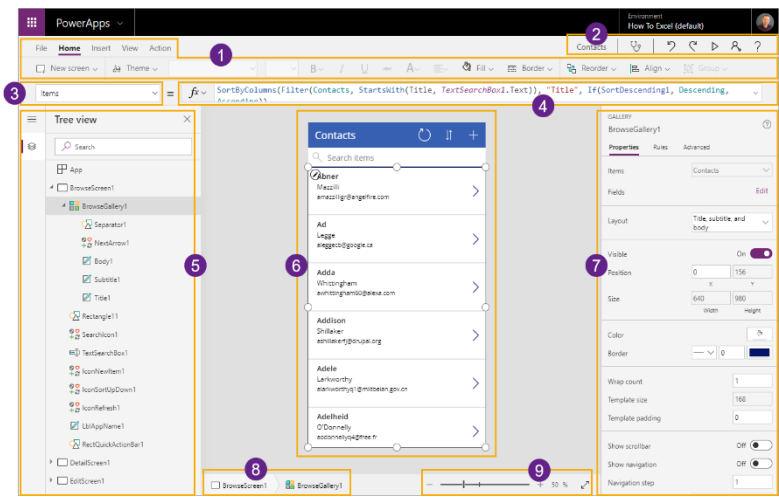
The two leading examples of metasoftware created by major technology companies that specialize in providing broad access cloud-based software are Microsoft’s Power Platform and Google’s AppSheet. Amazon’s Honeycode platform qualifies as Big Tech metasoftware, but its functionality is limited compared with Power Platform and AppSheet.

1. Microsoft Power Platform

Power Platform was initially released in 2018 and consists of an interconnected suite of functional software building tools that, when combined, constitute metasoftware. This is because Power Platform enables the creation of an extremely wide range of functional software. The primary components of the Power Platform are:

- **Power Apps:** a visual software development tool that enables users to create apps with standard UI elements, formula-driven logic, and connections to a wide variety of data sources. An indicative screenshot of the Power Apps interface is shown in Figure 7

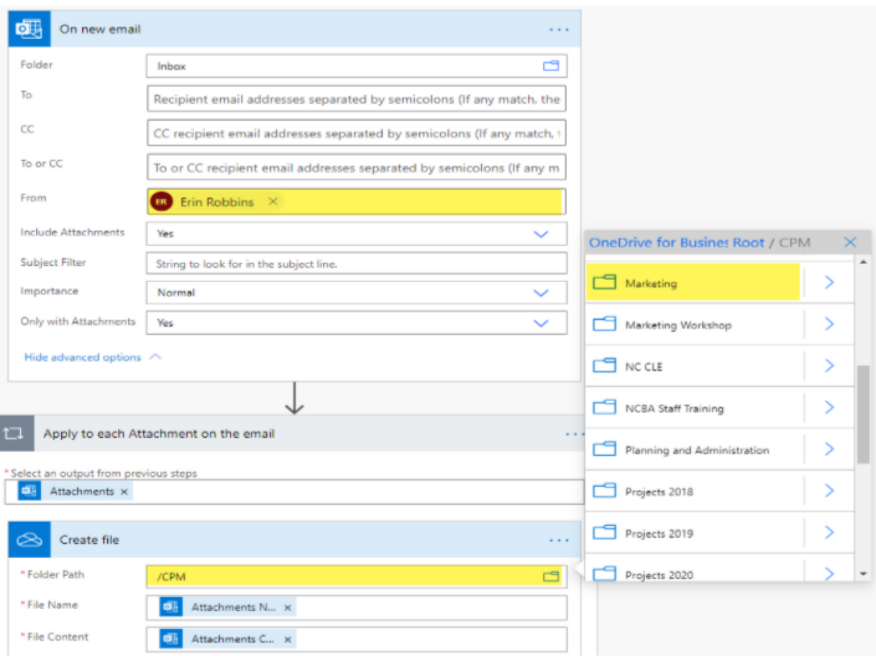
Figure 7: Power Apps Interface



The nine main components are

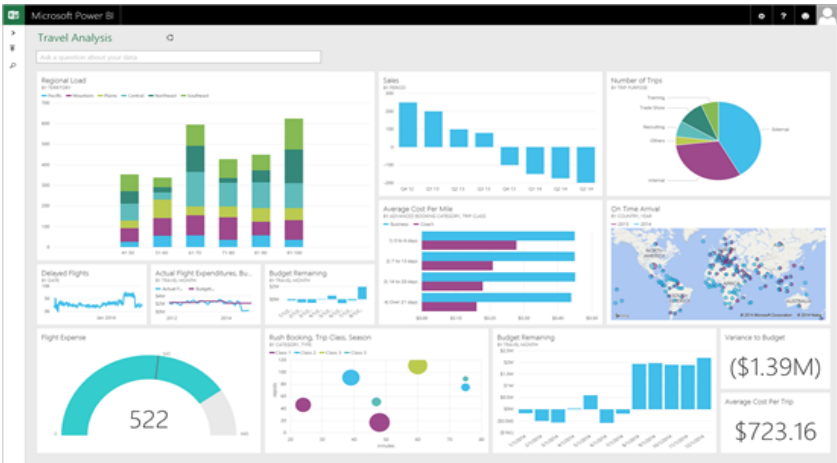
- Power Automate: a visual workflow automation tool. An indicative screenshot of the Power Automate interface that automates the process of saving email attachments to an online drive is shown in Figure 8.

Figure 8: Power Apps Interface



- Power BI: a data visualization and analytics tool. An indicative screenshot of the Power BI interface is shown in Figure 9

Figure 9: Power BI Interface



2. Google’s AppSheet

AppSheet was founded in 2014 and acquired by Google in January 2020.⁴⁰ AppSheet is not comprised of distinct products as is Power Platform, but it essentially combines the UI, workflow automation builder, and data connectivity into a single interface. Figure 10 shows the application editor with the UI menu.⁴¹

Figure 10: AppSheet

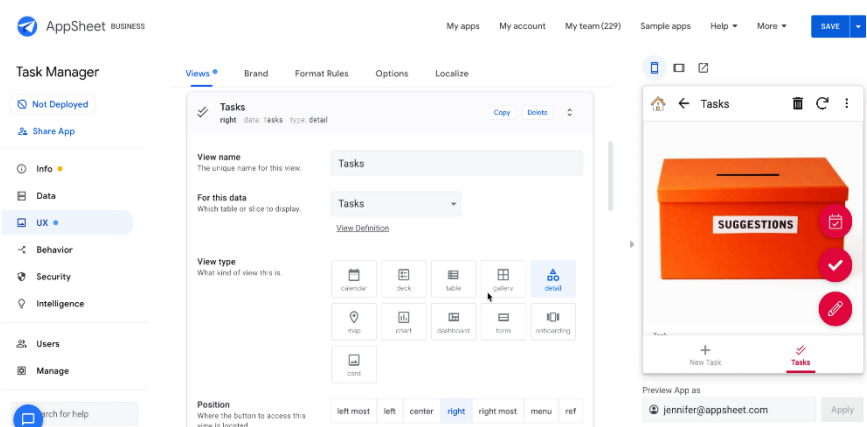
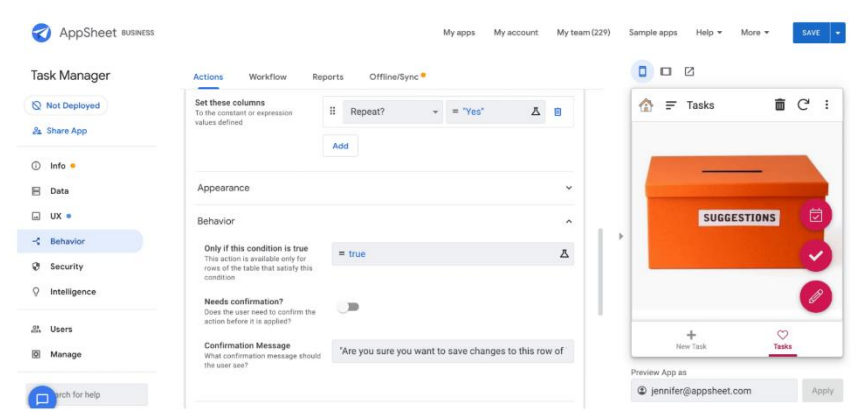


Figure 11 shows the same application editor but with a menu that enables a user to create programmable workflows.⁴²

⁴⁰ AppSheet, WIKIPEDIA, <https://en.wikipedia.org/wiki/AppSheet> [<https://perma.cc/HZU3-F25B>] (last visited Mar. 22, 2022).
⁴¹ How to Create an App, APPSHEET, <http://solutions.appsheet.com/how-to-create-an-app> [<https://perma.cc/4GVP-VT68>] (last visited Mar. 22, 2022).
⁴² Id.

Figure 11: AppSheet Workflow Editor

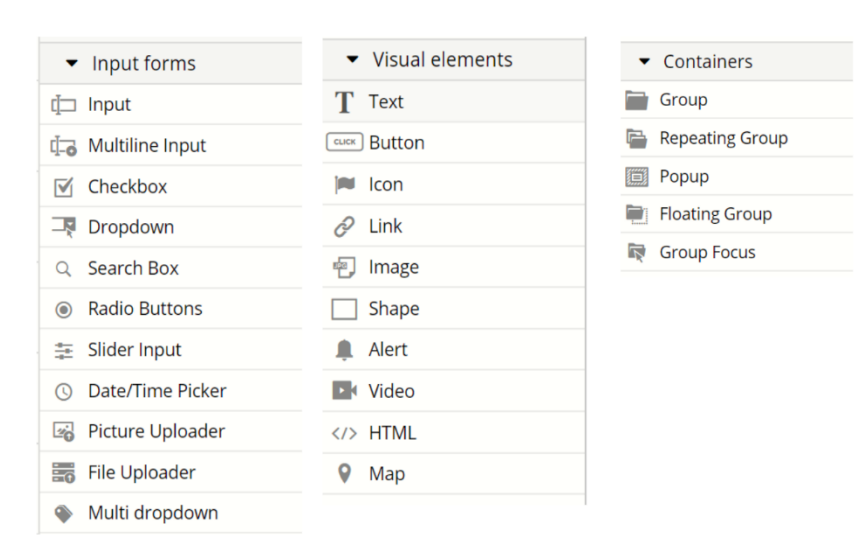


Like the Power Platform, apps built with AppSheet can connect to virtually any data source from popular databases such as Google Sheets and Excel, file storage services such as Dropbox, and enterprise data services such as Amazon Web Services and Oracle.

B. Standalone Proprietary Visual Software Platforms

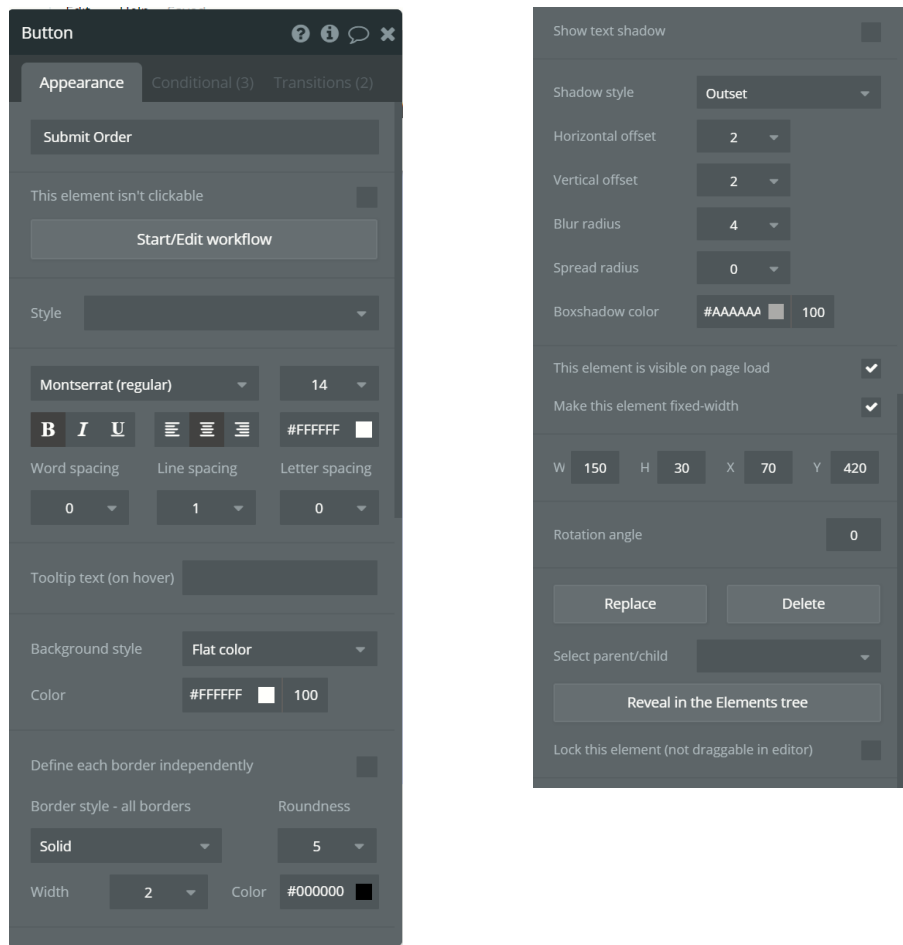
As shown in Figure 12, the standalone proprietary platform Bubble has a wide variety of built-in UI elements that can be added to a software application using the main application editor.

Figure 12: Bubble User Interface Editor



These elements can be added anywhere on a page by selecting, dragging, and dropping them. Once a UI element is added, its properties can then be viewed and modified by viewing and editing its corresponding property editor, an example of which is shown in Figure 13.

Figure 13: Bubble UI Element Property Editor



Bubble also enables users to display quantitative data from the application’s database or external sources in graphical form. This can be accomplished using a built-in graph UI element, a third-party plugin, or by embedding charts and other data visualization tools within an application using widely-used HTML and iFrame formats.⁴³ Bubble’s built-in chart element, for example, enables the display of line, bar, radar, pie, and doughnut charts.⁴⁴ Social science researchers can also use Bubble’s API plugin to interact with external data science and data visualization platforms.⁴⁵ This allows researchers to use Bubble to perform statistical analyses and present visual data.

Bubble enables visual programming of workflows using its Workflow builder. Figure 14 shows a very simple workflow example that

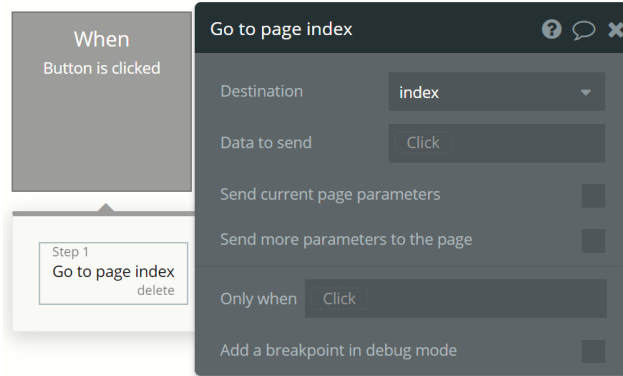
⁴³ *Introduction*, BUBBLE DOCS, <https://manual.bubble.io/#Data.Messages> [<https://perma.cc/MQ84-5P6S>] (last visited Mar. 22, 2022).

⁴⁴ *Id.*

⁴⁵ *Adding API Connections*, BUBBLE DOCS, <https://manual.bubble.io/building-plugins/adding-api-connections> [<https://perma.cc/LM4T-GDQ6>] (last visited Mar. 22, 2022).

programs software to navigate the application's user to the application's Index page if a certain button is clicked.

Figure 14: Simple Workflow in Bubble



Workflows can also implement programming logic on external software systems in order to integrate the data or functionality within an application built with Bubble. The standard approach for web applications to interact is using the RESTful API protocol.⁴⁶ The Bubble API plugin enables applications to connect to other applications. In a common REST-based action, an application in Bubble will send data to an external application service that will return data based upon the data that it receives from Bubble. For example, the Google Programmable Search Engine API enables an application built with Bubble to run a customized search on any specific website from within the Bubble application.⁴⁷ This is accomplished by sending search terms to the Google Search Engine API corresponding to the customized search.⁴⁸ In response, the Google API will send search results back to be displayed within the application created with Bubble.

Bubble enables data-oriented functionality for software application creators with a built-in database. In terms of database conceptual hierarchy, Bubble organizes databases, columns in a database, and rows in a database as types, fields, and things, respectively.⁴⁹ Developers can use workflows to connect UI elements to database-oriented workflows and schedule database actions when certain conditions are met. Bubble enables developers to program database actions using a visual expression builder that searches, displays, and edits database entries

⁴⁶ For an overview of RESTful APIs, See Kenneth Lange, *The Little Book on Rest Services*, Kenneth Lange (2016), <https://www.kennethlange.com/books/The-Little-Book-on-REST-Services.pdf> [https://perma.cc/Y26L-CYXU].

⁴⁷ *Programmable Search Engine*, GOOGLE, <https://programmablesearchengine.google.com/about/> [https://perma.cc/9PN5-NPDB] (last visited Mar. 18, 2022).

⁴⁸ *Using REST to Invoke the API*, GOOGLE, https://developers.google.com/custom-search/v1/using_rest [https://perma.cc/AWQ6-RZ2W] (last visited Mar. 18, 2022) (“Search query - Use the q query parameter to specify your search expression.”).

⁴⁹ *Bubble Key Concepts: types, fields and things*, BUBBLE GROUP, <https://bubblegroup.gitbooks.io/bubble-manual/content/working-with-data/key-concepts.html> [https://perma.cc/5XKV-SHE5] (last visited Mar. 18, 2022).

in numerous ways, such as sorting, combining, and formatting text.⁵⁰ Bubble enables different databases to refer to each other by enabling the field in one database to be single values or a list of values from the field of another database.⁵¹ Complex expression for extracting and replacing text in a database can also be used in Bubble with the formalized regular expression (or “RegEx”) approach.⁵²

C. Platform Independent Visual Software Platforms

Unlike the Big Tech and standalone metasoftware platforms analyzed above, a variety of metasoftware platforms have emerged that are built with fully open-source code, can be run on a variety of platforms, or at least code that is exportable by users. As discussed above, open-source and exportable code supports the metasoftware property of platform independence. Accordingly, the platforms discussed in this section rank the highest in terms of platform independence. However, because open-source software development tends to be more difficult to earn revenue due to its lack of proprietary value capture, open-source software tends to lag behind proprietary platforms in terms of the power of their enabling components.

One aspect of metasoftware platforms that have a significant amount of platform independence is that they enable users to decide *where* the functional software they create with the platform runs. The ability of users to choose where to run software is known as self-hosting. For example, as applied to the WordPress website builder:

[a] webmaster could choose to use either WordPress.com (“hosted”), which is a largely free service maintained by the company Automattic, or alternatively download the WordPress software from WordPress.org and install the WordPress software manually on their own private web server, whether that server is leased from a web hosting provider or on a server (physical hardware) owned by the webmaster (“self-hosted”).⁵³

Metasoftware platforms will typically describe their self-hosting capability with language such as “[y]ou are in control of all the source code, have full access to it and can use any hosting provider or a cloud platform for your websites.”⁵⁴ Metasoftware enables users to run the functional software they create on any platform and export the code

⁵⁰ *Introduction*, BUBBLE DOCS, <https://manual.bubble.io/#Data.Messages> [https://perma.cc/MQ84-5P6S] (last visited Mar. 18, 2022).

⁵¹ *Connecting Types With Each Other*, BUBBLE DOCS, <https://manual.bubble.io/working-with-data/connecting-types-with-each-other> [https://perma.cc/P9CB-DL8E].

⁵² *Operators & Comparisons*, BUBBLE DOCS, <https://manual.bubble.io/core-resources/data/operations-and-comparisons#extract-with-regex> [https://perma.cc/SQ36-PAUA] (last visited Mar. 18, 2022); *Regular expression*, WIKIPEDIA, https://en.wikipedia.org/wiki/Regular_expression [https://perma.cc/S83U-KHR6] (last visited Mar. 18, 2022).

⁵³ *About: Self-hosting (web services)*, DBPEDIA, [https://dbpedia.org/page/Self-hosting_\(web_services\)](https://dbpedia.org/page/Self-hosting_(web_services)) [https://perma.cc/Y4V8-LJSL] (last visited Mar. 18, 2022).

⁵⁴ *Unleash Creativity, Visual, Low Code Productivity Tools for Professionals*, WAPPLER, <https://wappler.io/> [https://perma.cc/4TGD-CNPA] (last visited Mar. 18, 2022).

underlying any functional software that is created. Flutterfly describes their functional-software export feature in terms of the options: “[v]iew the generated code in your browser, copy and paste it to your editor, download and run in your emulator, or connect it with GitHub in order to sync with your repository!” Draftbit’s Full Source Code Download feature enables users to “[e]xport [their] entire build as one clean, repo-ready package.”⁵⁵

The metasoftware platforms that enable users to create functional software that is the most platform independent are those whose metasoftware technology are completely open source. As noted above, when the metasoftware platform itself is open source, it is generally easier for the user to implement the code they export on alternative systems. Consistent with open-source software practice, the code that underlies metasoftware platforms is made freely available on the Internet using the standardized code repository Github.

The open source metasoftware platform, Saltcorn, enables users to build functional software applications such as blogs, dashboards, and project management tools. Saltcorn considers its primary benefit to be platform independence, which it describes in the following way:

There are many other application platforms with the same aims as Saltcorn, but very few of them are open source. For some people this won't matter, but the open source model will protect you from vendor lock in – being tied to one provider which may at any point raise its prices, cease to exist, discontinue features that you rely on. . . . It is a goal of Saltcorn that people who outgrow Saltcorn at some point in the future should also not be locked into the tool but be in a good place where they can take their database and build a Rails or Node application against it.⁵⁶

Consistent with my theory about metasoftware characteristic tradeoffs, Saltcorn is relatively weak in the substantive functionality that it enables and in its connectivity to external platforms. Saltcorn is primarily focused on enabling database oriented functional software, as opposed to software with strong front-end, graphical interfaces.⁵⁷ It also has no readily available integrations with external software systems, which requires them to be built from scratch.⁵⁸

⁵⁵ *Visual Building Done Right*, DRAFTBIT, <https://draftbit.com/features> [https://perma.cc/PV7D-89DZ] (lasted visited Mar. 16, 2022).

⁵⁶ *Frequently Asked Questions*, SALTCORN, <https://wiki.saltcorn.com/view/ShowPage?title=Frequently%20asked%20questions> [https://perma.cc/A8KJ-ZJ7D] (lasted visited Mar. 16, 2022).

⁵⁷ See Tom Nielsen, *Build with No Compromises, No Code, in No Time*, SALTCORN, <https://saltcorn.com> [https://perma.cc/JW2R-VQJ6] (last visited Mar. 16, 2022) (stating that “Saltcorn is a platform for building database web applications without writing a single line of code”).

⁵⁸ See *supra* note 58.

Another project that takes an open-source approach to creating a metasoftware platform is Plasmic.⁵⁹ However, because it still requires so much coding to build functional software, it is very weak on all three metasoftware characteristics. As noted in its documentation,

Plasmic as a page builder is its simplest and most common use case. Beyond pages, Plasmic can even be used to create frontends for complex web applications (such as Plasmic itself, which was built in Plasmic). This is possible because—despite being easy to start with—Plasmic gives you full visual control and works deeply with code. . . . For anything with non-trivial interactivity, all the logic/behavior—state bindings, event handlers, etc.—are all done from your own code.⁶⁰

D. Web3 and Decentralized Infrastructure

The most fundamental form of platform independence is the ability of software to run in any hosting environment and to be from immutable restrictions. For example, requiring users to share data with the hosting providers or be subject to content rules. The moniker given for this type of infrastructure is “Web3” -- so named for being the third major stage of progression of the internet. Web3 is often conceptualized as consisting of the following layers that serve a specific function:

- **state layer:** tamper-proof, universal data storage;⁶¹
- **computation layer:** means of carrying out computation that requires compatibility with one or more blockchain networks capability of carrying out the computation;⁶²
- **component layer:** digital assets such as tokens and identities;⁶³
- **protocol layer:** sets of standardized rules for activities such as token trading;⁶⁴
- **scalability/transfer layer:** additional networks that are required to improve the speed and functioning of the state layer;⁶⁵
- **user control layer:** provides the graphics for wallets and similar applications for users to initiate transactions and make other changes to the state layer;⁶⁶

⁵⁹ See generally *The Visual Builder for Your Tech Stack*, PLASMIC, <https://plasmic.app> [<https://perma.cc/BUB8-F2R7>] (last visited Mar. 17, 2022); *Welcome to Plasmic*, PLASMIC, <https://docs.plasmic.app/learn> [<https://perma.cc/X72Y-J8T5>] (last visited Mar. 17, 2022).

⁶⁰ *Id.*

⁶¹ Tekisalp, *supra* note 35.

⁶² *Id.*

⁶³ *Id.*

⁶⁴ *Id.*

⁶⁵ *Id.*

⁶⁶ *Id.*

- **application layer:** hosts functional software applications that run on a distributed network (often referred to as distributed applications, or “dApps”).⁶⁷

From the standpoint of the metasoftware characteristic of platform independence, Web3 is an improvement over Web2 because Web3 is decentralized, or at least substantially more so than Web2. There are many aspects to decentralization; in this context, it generally means that no person or entity has control over the data, protocols, and operations of software that runs on a network. This includes the purpose for which the software is used, and the content produced using the software. This is often described as permissionless innovation, whereby anyone can build and run software on the network so long as they have sufficient technical and economic capacity. Authorization by the provider of a network is not required as it acts somewhat like a public road or other utility. More technically, according to Javid Zarrin et al., Web3 decentralization via blockchain is achieved because

. . . each transaction through its peers in the network is done only by two nodes at a time and does not need a third-party validation. Decentralization allows the Blockchain to be non-reliant on a central authority. This enables nodes to essentially have equal voting rights within the network, which is then utilized with the consensus algorithm to dictate the Blockchain.⁶⁸

With respect to metasoftware platform independence, Web3 offers the highest degree of platform independence for any piece of functional software created on the platform. A decentralized software infrastructure means that functional software has the greatest degree of flexibility with respect to the nature and manner of how the software can run.

IV. BUILDING BLOCKS FOR LEGAL TECHNOLOGY

A. *LegalTech and Metasoftware Tradeoffs*

Metasoftware enables the creation of functional software, including legal technology software (“legaltech”). The various components of metasoftware accordingly serve as building blocks for legaltech.

Legaltech is software that contributes to the practice of law by furthering client objectives. The goals of legaltech include enhancing the delivery of legal services and furthering public interest goals and access to

⁶⁷ *Id.*

⁶⁸ Javid Zarrin, et al., *Blockchain for Decentralization of Internet: Prospects, Trends, and Challenges*, CORNELL U. arXiv:2011.01096, Nov. 2, 2020, at 7, <https://arxiv.org/pdf/2011.01096.pdf#:~:text=Decentralization%20allows%20the%20Blockchain%20to,be%20validated%20by%20trusted%20miners> [https://perma.cc/T3BU-X4RX] (last visited Mar. 24, 2022).

the legal system.⁶⁹ Legaltech has many subcategories that mirror those of legal practice areas and the application of particular technologies. Given the commercial potential of algorithmic law, legaltech often seeks to automate legal reasoning to make deterministic conclusions about the application of law based upon user information and other facts.

Major general categories of legaltech include:

- legal research
- electronic discovery
- automated pleading, motion, and discovery document-writing and analysis
- lawyer-client matching platforms
- law firm matter management
- litigation analytics
- contract management and drafting automation
- digitally-assisted and automated contract negotiations
- digitizing contracts and other documents into digital-native, structured-data documents
- contract analysis and analytics
- corporate legal operations analytics and workflow automation
- legal intake and analysis via expert systems
- online dispute resolution.

Legaltech applications utilize a wide variety of technologies that are common to modern applications such as artificial intelligence and automation.⁷⁰ For example, legaltech developers create software to analyze case briefs to make suggestions for improvement, compare hundreds of thousands of contracts to produce quantitative risk assessments, and predict the rulings of judges. Digitized documents, data about documents, and legal services data can also be incorporated into an enormous range of semi and fully automated digital workflows that

⁶⁹ Marc Lauritsen & Quinten Steenhuis, *Substantive Legal Software Quality: A Gathering Storm?*, INT'L CONF. OF A.I. AND L. (2019), <https://dl.acm.org/doi/pdf/10.1145/3322640.3326706> [<https://perma.cc/LCT3-QWD8>] (last visited Mar. 24, 2022) (stating that “Automated legal services may be the best hope for access to justice and legal wellness for billions of our fellow humans. AI & Law activists are encouraged to find ways to bring their utensils to the feasts of knowledge automation that lie ahead”).

⁷⁰ Daniel Faggella, *AI in Law and Legal Practice – A Comprehensive View of 35 Current Applications*, EMERJ.COM (Sept. 7, 2021), <https://emerj.com/ai-sector-overviews/ai-in-law-legal-practice-current-applications/> [<https://perma.cc/ZH3A-EPZQ>] (last visited Mar. 24, 2022); Lauri Donahue, *A Primer on Using Artificial Intelligence in the Legal Profession*, JOLT DIGEST (Jan. 3, 2018), <https://jolt.law.harvard.edu/digest/a-primer-on-using-artificial-intelligence-in-the-legal-profession> [<https://perma.cc/MAW5-RBP6>] (last visited Mar. 24, 2022); Bernhard Walit & Roland Vogl, *Explainable Artificial Intelligence – the New Frontier in Legal Informatics*, JUSLETTER IT 22 (2018), https://jusletter-it.weblaw.ch/services/login.html?targetPage=http://jusletter-it.weblaw.ch/issues/2018/IRIS/explainable-artifici_fbce1ac1d0.html__ONCE&handle=http://jusletter-it.weblaw.ch/issues/2018/IRIS/explainable-artifici_fbce1ac1d0.html__ONCE [<https://perma.cc/E5NZ-6FCP>] (last visited Mar. 24, 2022) (identifying numerous different types of AI reasoning approaches in law legal scholarship); *IACCM-Capgemini Automation Report*, HUBSPOT (Apr. 12, 2019), https://s3.eu-central-1.amazonaws.com/iaccmportal/resources/files/10162_iaccmcapgeminautomationreport.pdf [<https://perma.cc/K39X-JVC5>] (last visited Mar. 24, 2022) (providing an overview of contract automation tools).

include client-related services and decision making. In terms of user interface, legaltech apps may use traditional point and click, touch screens, voice inputs, and interactive chat bots.

B. Metasoftware Tradeoffs

The tradeoffs for the different types of metasoftware platforms identified in Table 1 in Section III above have important implications for using metasoftware to build legaltech. This is because different types of legaltech are dependent on different metasoftware characteristics, such as enabling components or connectivity. The following subsections discuss how metasoftware can lay the basis as building blocks for major categories of legaltech and any tradeoffs or other challenges involved.

C. Legal Research

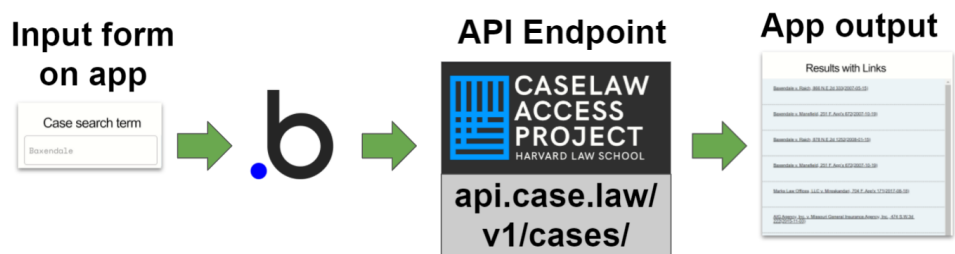
Computational capabilities have long been applied to researching the relevant primary and secondary sources applicable to a given legal issue. Traditionally, legal research has relied upon using proprietary platforms that digitally organize, archive, and update legal research materials such as case law, statutes, and regulations. More recently, primary and secondary legal research materials have become more readily available online, often as a free public resource.

Accordingly, from a metasoftware building blocks perspective, the most important aspect for legal research is being able to create a user interface that displays the results of legal research. What is also important is the connectivity aspect of metasoftware that can connect to legal research material that is available external to the metasoftware platform through an API.

For example, the standalone proprietary metasoftware platform Bubble could be used to create functional legal research software by connecting to Harvard's Caselaw Access Project, which provides 6.7 million unique cases for free via API.⁷¹ A user would insert case law search terms using an input user interface element. That search term would be sent to the Caselaw Access Project's database of cases and retrieve the relevant search results. This overall workflow is depicted in Figure 15:

⁷¹ *Caselaw Access Project*, HARV. L. SCH., <https://case.law/> [<https://perma.cc/3EJ7-UT8P>] (last visited Mar. 24, 2022).

Figure 15: Using Bubble to Create Legal Research Software



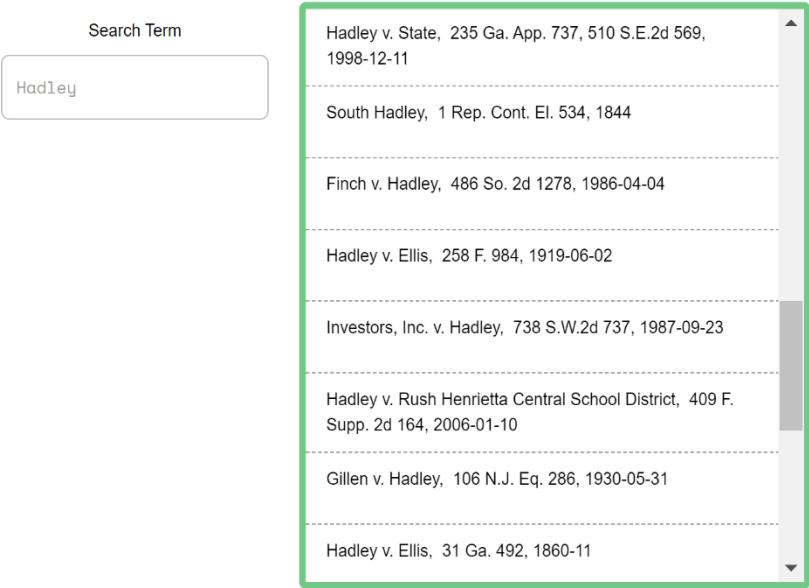
When the Bubble app is connected to the Caselaw Access Project and a search term is sent to the Project, the Bubble API connector will show the exact same results as API’s results in JSON. Figure 16 shows how Bubble’s API Connector draws in search results from the Caselaw Access Project API.

Figure 16: Bubble API Connector Drawing in Caselaw Access Project Data



Accordingly, metasoftware can be used to create functional legal research software out of two building blocks: 1) an API connection to the Caselaw Access Project, and 2) the user interface elements of a search term input and a search results output list.

Figure 17: Functional Legal Research Software Created with Metasoftware Platform



These two building blocks can serve as the basis for additional functionality, such as enabling users to save and sort their search results, adding additional legal resources to search, and applying search algorithms to improve search results.

D. Matter Management

A foundational application of legaltech in practice is matter management. A standard industry definition of matter management is:

Matter management is the process of gathering, tracking, assigning and reporting on legal work including matter name, type, legal service providers and in-house counsel working on the case, budgets and invoices. A matter can be a simple task requested of corporate counsel, or a complex legal project. For example, legal research, contracts, disputes and litigation, intellectual property or [mergers and acquisitions] M&A. Matter management enables more effective organization, collaboration and reporting of a legal department’s work and associated costs.⁷²

Other important functionalities of matter management include being able to “open matters of various types, assign budgets and billing

⁷² *What is Legal Matter Management?*, BUSYLAMP, <https://www.busylamp.com/what-is-legal-matter-management/> [https://perma.cc/9QLX-2KUB] (last visited Mar. 24, 2022); see also *What is Legal Matter Management? And Why it Matters to Legal Ops*, MIRATECH (Aug. 13, 2020), <https://mitratech.com/resource-hub/blog/legal-matter-management/> [https://perma.cc/L5YX-AVVM].

guidelines, assign timekeepers to matters and sub-tasks, and report on who worked on what, for how long.” According to a 2015 Thomson Reuter survey, users of matter management tools most commonly expect the following functionality:

- Integration with Outlook: email, calendars, etc.
- Automate tasks whenever possible (e.g. produce standard letters and legal forms)
- Schedule tasks
- Place tasks to be completed in fee earners electronic diaries
- Evaluate tasks when they are overdue
- Integration with time recording and billing functionality.⁷³

Given the prevalence of Microsoft Outlook for in-house legal departments and law firms, the optimal building blocks for legaltech matter management are those that make up the Microsoft Power Platform.⁷⁴ At a high level, the Power Platform’s products can be used for matter management in the following way:

- Word documents, PDF documents, copies of email, and other files can be stored on OneDrive, organized in folders and subfolders, and labeled and tracked as appropriate.
- Various sharing and permission options of files within OneDrive such as within SharePoint or Teams.
- Organizing files, syncing calendars, obtaining approvals, and producing matter management information by applying Power Automate to Outlook emails or documents in OneDrive, including Excel spreadsheets that hold matter management relevant data.
- Applying Power BI for more sophisticated insights about performance and optimal resource allocation.
- Using SharePoint to create and store metadata about documents and matters for search and analytics.
- Integrating with external legal billing services.⁷⁵

As an example of how the Microsoft Power Platform is used for matter management, consider how OneDrive enables files and folders to be marked however the user chooses, and this label can correspond to a status under a legal matter management method of organization, such as case or deal status. As shown in the following figure, unrelated documents can be grouped together by something as simple as a “Saved for later”

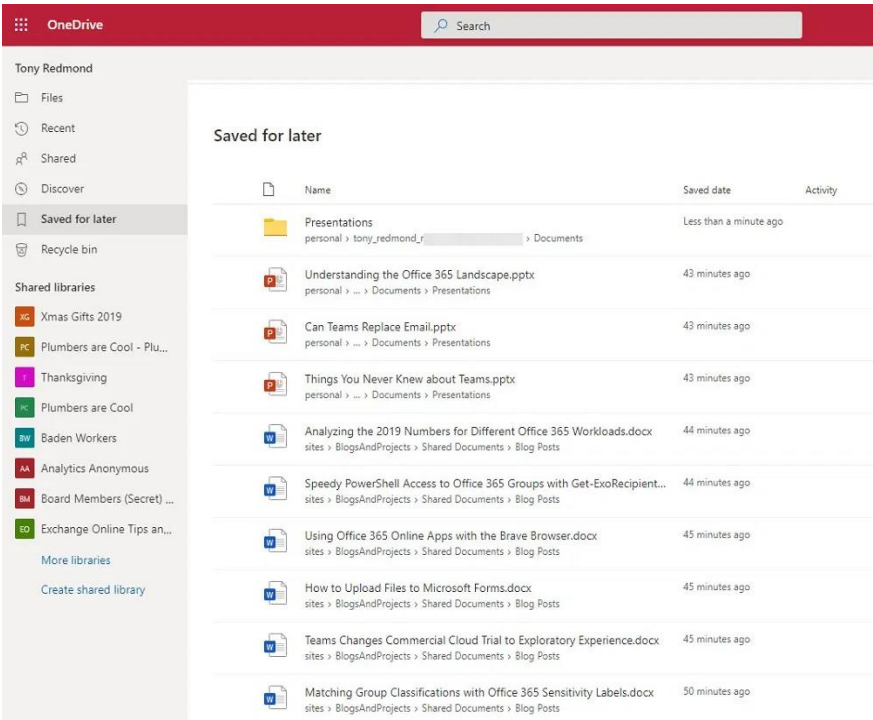
⁷³ Jeffery, Chris, *Focus on Matter Management for Small Law Firms*, THOMSON REUTERS, <https://www.thomsonreuters.com/content/dam/openweb/documents/pdf/legal/white-paper/matter-management-in-law-firms-product.pdf> [https://perma.cc/8B2X-9V7J] (last visited Mar. 24, 2022).

⁷⁴ 7 *Indisputable Reasons to Use Office 365 for Matter Management*, REPSTOR, <https://www.repstor.com/wp-content/uploads/2017/09/O365-for-Matter-Management-paper-Final-Sept-17.pdf> [https://perma.cc/BPG5-9WVF] (last visited Mar. 24, 2022) (noting that “one in five corporate employees now has Office 365, making it the most widely used enterprise cloud service”).

⁷⁵ See generally, *supra* note 75.

status to alert the user of ongoing tasks with respect to the incomplete document.

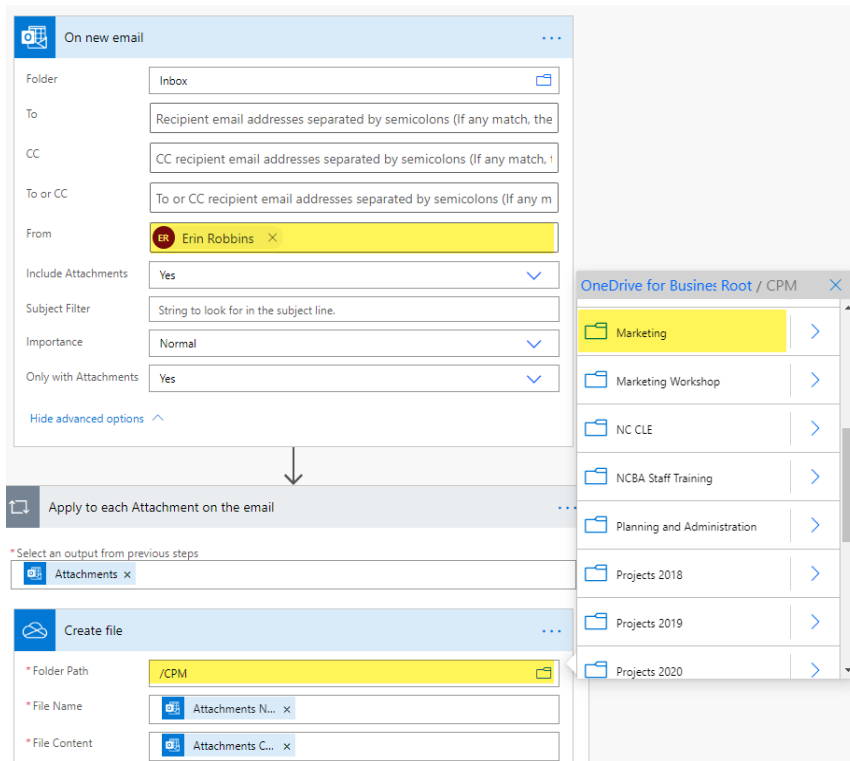
Figure 18: OneDrive Organization



Another example of Power Platform’s use in matter management is using Power Automate to automate the process of saving and organizing email attachments in OneDrive. Further customizations to Power Automate include saving emails in a particular folder when received from a specific person and updating internal and external collaborators in a Teams channel that a new email has arrived from that person.⁷⁶ The following figure shows a Power Automate workflow automation that saves only emails from a person named Erin Robbins in a particular OneDrive folder.⁷⁷

⁷⁶ See Reach, Catherine, *Microsoft 365 Power Automate*, NORTH CAROLINA BAR ASSOCIATION CENTER FOR PRACTICE MANAGEMENT, EMAIL MANAGEMENT, MICROSOFT OFFICE, PRODUCTIVITY (June 23, 2020), <https://www.ncbar.org/2020/06/23/microsoft-365-power-automate/>, [https://perma.cc/7B7D-DFYG].

⁷⁷ See *id.*

Figure 19: Power Automate Workflow

E. Contract Automation

Contract automation technology digitizes and automates aspects of the creation, management, and analysis of legal agreements throughout their lifecycle, from inception to termination of the relationship between parties. According to an IACCM-Capgemini Report, contract automation technology has 14 general capabilities:

- Contract drafting
- Contract approvals
- Contract query
- Contract discovery
- Obligation management
- Document repository
- Dispute management
- Performance management and calculations
- Contract change management
- Contract information extraction (machine learning)
- Collaboration with counterparties
- Requests for proposals
- Management reporting

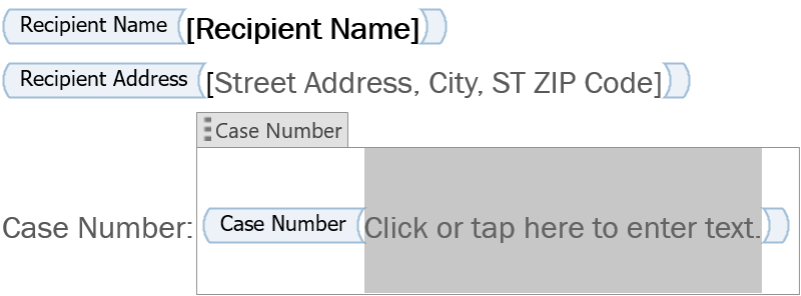
- Contract portfolio analysis⁷⁸

Components of metasoftware can be used as the building blocks for functional contract automation software. Given the ubiquity of Microsoft Word in drafting contracts and PDF format for official versions of executed documents, the Microsoft Power Platform is the most appropriate platform for constructing the building blocks of functional contract automation software.

A foundational aspect of contract automation is drafting contracts using pre-established and pre-approved templates instead of drafting from scratch and seeking approval for each new agreement. Using the Microsoft Power Platform to enable users to populate contract templates entails the following general steps:

1. Create a template in Word by adding variable fields (called “content controls”) within the document that will serve as placeholders for text that is populated with data.
 - An example of a template for a form letter with fields for Recipient Name, Recipient Address, and Case Number is as follows:

Figure 20: Example Word Template



2. Trigger the creation of a new document from a template by connecting the template to a Microsoft Power Automate workflow that triggers the population of the template with data from a spreadsheet.
 - An example of populating the foregoing template with a manually triggered Power Automate workflow is shown in the Figure below:

⁷⁸ IACCM-Capgemini, *supra* note 75.

Figure 21: Manually Triggering Template Data Population with Power Automate

Populate a Microsoft Word template

* Location: SharePoint Site - DocGen

* Document Library: Templates

* File: /Case Letter Template.docx

Recipient Name: Recipient Name x

Recipient Address: Address x

Case Number: Case Number x

Sender Name: Sender Name x

Sender Title: Sender Title x

Add dynamic content +

- 3. Define what action to take after the new document has been created from the form, such as saving the new document to a particular folder.⁷⁹

F. Artificial Intelligence Applications

Given the importance and growing application of the machine learning subdomain of artificial intelligence, this section will focus on AI metasoftware and how the building blocks of AI for legaltech can be constructed.

In order for metasoftware platforms to serve as a blank slate, they must enable users to build state-of-the-art software functionality. This generally occurs when metasoftware platforms integrate with more specialized software providers and builders, such as those that use or enable artificial intelligence, augmented and virtual reality, and gaming. Blank-slate platforms should accordingly be conceptualized with respect to these more specific functionalities such as platforms that enable users to build a vast variety of artificial intelligence software.

The primary concepts in AI are defined as follows:

“Artificial Intelligence” is the term used to describe how computers can perform tasks normally viewed as

⁷⁹ See generally, *Use Word templates to create standardized documents*, MICROSOFT, (Feb 15, 2022), <https://docs.microsoft.com/en-us/power-platform/admin/using-word-templates-dynamics-365>, [https://perma.cc/48B9-B63X]; Bernier, Hugo, *Generate Word documents from a template using Power Automate*, TAHOE NINJAS (March 13, 2020), <https://tahoeninjas.blog/2020/03/13/generate-word-documents-from-a-template-using-power-automate/> [https://perma.cc/7E52-USEA].

requiring human intelligence, such as recognizing speech and objects, making decisions based on data, and translating languages. AI mimics certain operations of the human mind.

"Machine learning" is an application of AI in which computers use algorithms (rules) embodied in software to learn from data and adapt with experience.⁸⁰

[A] neural network is [an algorithm used to make decisions] comprised of four main components: inputs, weights, a bias or threshold, and an output.⁸¹

Deep learning uses a "neural network that consists of more than three layers."⁸²

A variety of AI metasoftware platforms that essentially serve as a blank slate on which to build functional legaltech software already exist. Several are summarized as follows:

- Akkio is an end-to-end AI platform that enables users to integrate AI into workflows with a visual interface. Akkio enables users to build and deploy machine learning neural network models from data using visual "flows." These models are used to predict an outcome (or output) that the user chooses. For example, 80% of the data may be used to create a model, and the other 20% is used to validate the model (i.e., check how well it works). Akkio also enables users to create a simple mobile or web application using the model that takes in new data and gives the user an output (prediction) based on the model that was created.⁸³
- Peltarion is a platform for building both supervised and unsupervised machine learning models with a similar approach to building models off data, evaluating the efficacy of the model, and deploying the model in an application. Peltarion enables a wide variety of AI applications such as building image classifiers, text classifiers, image and text similarity models, multi-label classifications, image segmentation, and natural language processing to classify text. The following figure shows how "blocks" are used in Peltarion, which "represent the

⁸⁰ Donahue, Lauri, *A Primer on Using Artificial Intelligence in the Legal Profession*, JOLT DIGEST (Jan 3, 2018), <https://jolt.law.harvard.edu/digest/a-primer-on-using-artificial-intelligence-in-the-legal-profession> [https://perma.cc/JB3M-BU75].

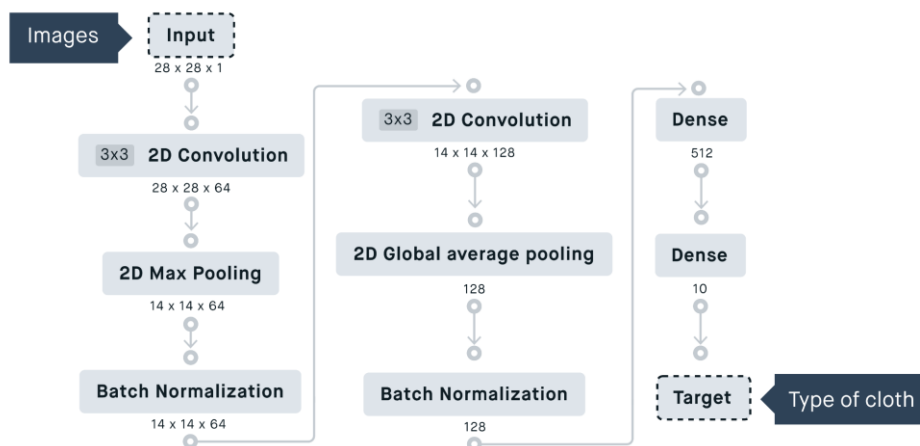
⁸¹ Kavlakoglu, Eda, *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?*, IBM (May 27, 2020), <https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks> [https://perma.cc/6ZNE-SLKQ].

⁸² *Id.*

⁸³ See Akkio, *Getting Started With Akkio*, YOUTUBE (Dec. 4, 2020), https://www.youtube.com/watch?v=5V4o_gcIfx0 [perma.cc/24M8-UL6W].

basic components of a neural network and/or the actions that can be carried out on them.”⁸⁴

Figure 22: Representative Machine Learning Algorithm Components and Actions



- Monkeylearn is another AI metasoftware platform that enables users to use AI to engage in (1) sentiment analysis, (2) data cleaning, (3) sorting data that has not been structured according to a predefined system, (4) machine learning as referred to above, (5) natural language processing, (6) topic analysis, (7) keyword extraction, (8) text classification.⁸⁵

These AI metasoftware platforms can be used to create the building blocks of fully functional AI-powered solutions. According to a comprehensive study of how AI is used in legaltech, its application falls into six categories:

- Due diligence – Litigators perform due diligence with the help of AI tools to uncover background information. . .
- Prediction technology – An AI software generates results that forecast litigation outcome[s].
- Legal analytics – Lawyers can use data points from past case law, win/loss rates and a judge’s history to be used for trends and patterns.
- Document automation – Law firms use software templates to create filled out documents based on data input.
- Intellectual property – AI tools guide lawyers in analyzing large IP portfolios and drawing insights from the content.

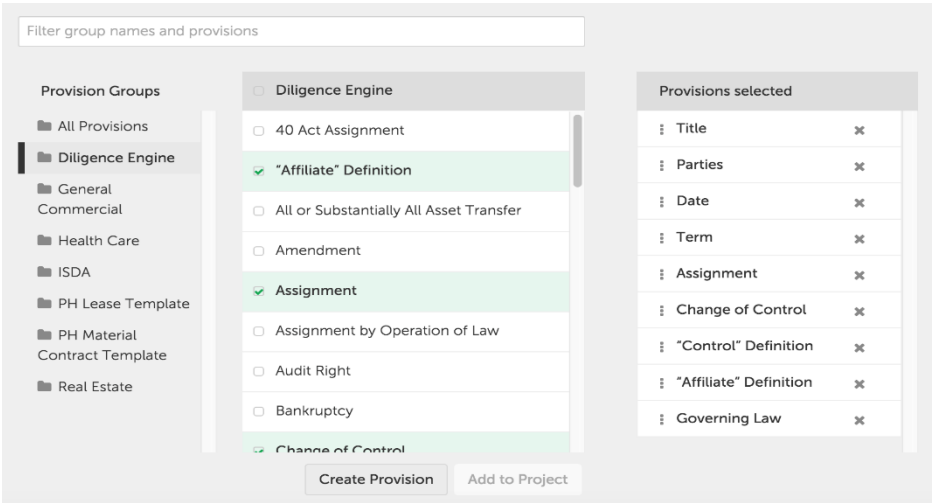
⁸⁴ *Blocks*, PELTARION, <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks> [perma.cc/F7DK-GUV3] (last visited Mar. 10, 2022).

⁸⁵ *AI Guides*, MONKEYLEARN, <https://monkeylearn.com/guides/> [perma.cc/NM5D-2WKQ] (last visited Mar. 10, 2022).

- Electronic billing – Lawyers’ billable hours are computed automatically.⁸⁶

Due diligence relies on large-scale, rapid AI-driven text extraction, comparison, classification (e.g., high-risk language), quantification (e.g., levels of enforceability), and summarization.⁸⁷ Some systems claim to be able to review over 10,000 documents in seconds.⁸⁸ The following figure shows a screenshot from Kira’s due diligence functionality that organizes definitions throughout agreements to enable lawyers to compare them for consistency.⁸⁹

Figure 23: AI-Driven Due Diligence



AI can also assist in both the transactional and litigation document drafting by suggesting proper language or edits based upon data such as what language is more likely to be accepted or challenged or what language judges may find more persuasive.⁹⁰ Functional software provider Blackboiler describes its AI-driven software that suggests edits in the following way:

BlackBoiler is the only patented, 100% AI-powered tool that learns from your company’s playbook and your edits to previously reviewed company documents to instantaneously review and markup contracts in track changes, just like an attorney.⁹¹

⁸⁶ Faggella, Daniel, *AI in Law and Legal Practice – A Comprehensive View of 35 Current Applications*, EMERJ, <https://emerj.com/ai-sector-overviews/ai-in-law-legal-practice-current-applications/> [perma.cc/8SX5-EP4U] (Sept. 7, 2021).
⁸⁷ *Id.*
⁸⁸ Morgan claims that their program, named COIN (short for Contract Intelligence), extracts 150 attributes from 12,000 commercial credit agreements and contracts in only a few seconds.
⁸⁹ *Due Diligence*, KIRA, <https://kirasystems.com/how-kira-works/due-diligence/> (last visited Mar. 10, 2022) [perma.cc/F49S-EAUY].
⁹⁰ EMERJ, *supra* note 91.
⁹¹ *Automated Contract Review*, BLACKBOILER, <https://www.blackboiler.com/automated-contract-review/> (last visited Mar. 10, 2022).

Accordingly, functional AI software can be trained to assist attorneys in performing tasks such as editing contracts according to previously used approaches and undertaking trademark and patent applications. In electronic discovery, AI systems can review how previous documents were scored by lawyers as being relevant to litigation and then perform its own review and marking of documents.⁹²

IV. CONCLUSION

Metasoftware is a fundamentally new category of software that has no functionality per se except to enable users to build and customize functional software. Metasoftware is a new category because, unlike traditional functional software, it can enable users to create the widest possible range of software functionality, integrate with external data and systems, and run in any computing platform without being tied to any particular environment.

There are three types of metasoftware platforms: large scale cloud-providers, standalone proprietary platforms, and open-source platforms. Each of these types of metasoftware platforms differs with respect to the essential characteristics of metasoftware:

- Cloud provider-based platforms have high levels of enabling properties and connectivity for their complementary cloud-based products but are inherently low with respect to platform independence due to their business model.
- Standalone proprietary platforms have high levels of enabling properties and connectivity for their plugin ecosystem but are inherently low with respect to platform independence due to their business model.
- Open-source platforms have low to medium levels of enabling properties and connectivity but, by definition, the highest levels of platform independence as their business model does not rely on capturing users for cross-selling products or platform usage fees.

Based on the relative strengths and weaknesses with respect to building functional software for each type of metasoftware platform, each platform is generally best suited to building different types of legaltech. Based on the discussion in this article, the type of metasoftware best suited for building functional legaltech applications is as follows:

- Standalone proprietary metasoftware platforms are best for legal research because of their high and accessible connectivity to online primary and secondary legal research data.

⁹² EMERJ, *supra* note 91.

- Microsoft's cloud-based Power Platform is best suited for legal matter management and contract automation technology and relies heavily on other Microsoft products such as Outlook and Word.
- Cloud provider and standalone metasoftware platforms are best suited for integrating artificial intelligence metasoftware due to the high connectivity of cloud provider and standalone platforms.