Flynn Allen J (Orcid ID: 0000-0002-3471-3063)

CBK Model Composition Using Paired Web Services and Executable Functions: A Demonstration for Individualizing Preventive Services

Adam Beck, Department of Learning Health Sciences, Medical School University of Michigan
adambeck@umich.edu

Peter Boisvert, Department of Learning Health Sciences, Medical School University of Michigan
pboisver@umich.edu

Philip Boonstra, School of Public Health, University of Michigan
philb@umich.edu

Tanner Caverly, Department of Learning Health Sciences, Medical School University of Michigan
tcaverly@umich.edu

Nate Gittlen, Department of Learning Health Sciences, Medical School University of Michigan
nathaniel.gittlen@gmail.com

George Meng, Department of Learning Health Sciences, Medical School University of Michigan
gqmeng.meng@gmail.com

Brooke Raths, Department of Learning Health Sciences, Medical School University of Michigan
braths@med.umich.edu

Glen Taksler, Cleveland Clinic
takslerg@ccf.org

Charles P. Friedman, Department of Learning Health Sciences, Medical School University of Michigan
cpfried@umich.edu

Corresponding author:
Allen Flynn, Department of Learning Health Sciences, Medical School University of Michigan
ajflynn@umich.edu
734.223.7483

Abstract

Introduction

Learning health systems are challenged to combine computable biomedical knowledge (CBK) models. Using common technical capabilities of the World Wide Web (WWW), digital objects called Knowledge Objects, and a new pattern of activating CBK models brought forth here, we aim to show that it is possible to compose CBK models in more highly standardized and potentially easier, more useful ways.

Methods

Using previously specified compound digital objects called Knowledge Objects, CBK models are packaged with metadata, API descriptions, and runtime requirements. Using open-source runtimes and a tool we developed called the KGrid Activator, CBK models can be instantiated inside runtimes and made accessible via RESTful APIs by the KGrid Activator. The KGrid Activator then serves as a gateway and provides a means to interconnect CBK model outputs and inputs, thereby establishing a CBK model composition method.

Results

To demonstrate our model composition method, we developed a complex composite CBK model from 42 CBK submodels. The resulting model called CM-IPP is used to compute life-gain estimates for individuals based their personal characteristics. Our result is an externalized, highly modularized CM-IPP implementation that can be distributed and made runnable in any common server environment.

Discussion

CBK model composition using compound digital objects and the distributed computing technologies is feasible. Our method of model composition might be usefully extended to bring about large ecosystems of distinct CBK models that can be fitted and re-fitted in various ways to form new composites. Remaining challenges related to the design of composite models include identifying appropriate model boundaries and organizing submodels to separate computational concerns while optimizing reuse potential.

Conclusion

Learning health systems need methods for combining CBK models from a variety of sources to create more complex and useful composite models. It is feasible to leverage Knowledge Objects and common API methods in combination to compose CBK models into complex composite models.

Keywords: computable biomedical knowledge, model composition, decentralized web technology

INTRODUCTION:

In 2009, Tsafnat and Coiera discussed several challenges related to reasoning across multiple biomedical models[1]. They highlighted the challenges of computer-aided model construction, automated model selection, and model composition. This paper focuses on model composition, which is the process of building up better reasoning capabilities by connecting or combining multiple models to form *composite models*[1,2].

The topic of model composition is not new, but it is timely[3-6]. In biomedicine, there is growing evidence from fields including whole-cell modeling and integrated systems biology that composite models can improve our understanding of biology and human health[6,7]. It is now conceivable that model composition could become central to a lot of future scientific work in biomedicine[8]. Hence, for learning health systems, model composition seems vital[9].

Throughout this paper, "model" refers to computer-processable implementations of results and insights previously revealed through empirical scientific inquiry and learning. In these types of models, such results and insights are expressed concretely and formally as conceptual, logical, mathematical, or statistical statements about variables and the relationships between variables, including causal and correlative relationships[10].

This paper contributes a new method for building composite models by connecting them via their inputs and outputs. We recognize that model composition in software is nothing new. Software that brings many models together by interrelating the inputs and outputs of discrete functions has been around for a long time. What is new here is that how the models we combine are individually externalized and modularized using compound digital objects called Knowledge Objects (KOs). We have previously published our Knowledge Object Reference Ontology (KORO) which describes the parts and pieces of Knowledge Objects in detail[11]. Following KORO, the KOs for this study enable models to be treated both as static resources and active web services.

Using multiple KOs, we are primarily interested in composing models of *computable biomedical knowledge* (CBK). CBK models may also be called CDS artifacts or machine learning, deep learning, AI, decision, and business process models, or even *actionable knowledge units*[12-14]. Here we generally refer to any model that represents biomedical results and insights as a *CBK model*. To better support learning health systems, we demonstrate the building of *composite CBK models* by interconnecting multiple distinct CBK submodels packaged inside many individual KOs.

This model composition work is generally motivated by three main drivers relevant to learning health systems. First, the relational nature of knowledge calls for connections between disparate results and insights[15-17]. Second, the acceleration of scientific activity and attendant accumulation of new results and insights increase the need to connect new and prior knowledge to extend and apply what is learned[18]. Third, different types of knowledge exist and necessarily have dissimilar computer-processable representations[19-21]. Hence, to advance biomedical science, enable learning health systems, and improve human health, more distinct computable models represented variously need to be connected or combined effectively than in the past[22].

Research and development on decentralized web technology for model reuse strongly influences our approach[23]. Web app developers will be familiar with our approach since they are accustomed to building web applications with reusable *software libraries* or *packages*, some of which contain "models" per our definition of the term[24]. Similarly, data scientists perform model composition using tools and languages that strongly support code reuse, such as Machine Learning in Julia (MLJ)[2]. In addition, to compute with models represented variously, polyglot virtual machines supporting a wide variety of *runtimes* (or software execution engines) enable model composition of submodels encoded in different programming languages and formats[25].

Our CBK model composition method ultimately relies on connecting pairs of web services and corresponding executable functions backing the web services. We apply our method to connect and combine preventive medicine models into a composite model with 42 submodels to support *individualized precision prevention* (IPP). In the end, the composite model we produce computes individualized estimates of life gain for 21 different evidence-based preventive services[26].

Next, in the Methods section, we begin by outlining our general technical approach to model composition. After that, we detail key technical items and give examples. Later, in the Results section, we describe the IPP composite model we developed with our methods and explain its use in an initial study of preventive service practices. Finally, we discuss our progress in the Discussion section and reflect on some key remaining challenges before concluding.

METHODS:

GENERAL TECHNICAL APPROACH TO MODEL COMPOSITION

Our primary goal is to support both the developers and end-users of end-user or *client applications* by giving developers and end-users ready access to powerful composite CBK models. As portrayed in Figure 1 below, our approach uses a stack of technical components for managing and deploying KOs, which are digital packages holding CBK models[27]. In Figure 1, the two yellow-shaded areas are where we make new technical contributions.

In the Server Layer of our technology stack just below the client applications layer, we rely on established World Wide Web (WWW) network components to handle standard HTTP requests and responses to and from lower-level microservices. In the Microservices Layer, custom microservice tools from our team, especially the Knowledge Grid (KGrid) Activator[27], organize, mobilize, and instantiate CBK models to get them running and make them network accessible.

In the lowest static CBK Layer of our stack, we specify the structure and contents of modular Knowledge Objects[11,28]. Each KO is stored as an individually identifiable package that bundles a distinct CBK model along with some other essential content described below. KOs can then be used for computing with assistance from the KGrid Activator tool.

Our stack brings in well-supported runtimes built by others for executing the CBK models packaged in KOs (Figure 1). To date, we have used runtimes for JavaScript (e.g., V8), Python, and R. Theoretically, there is no limit to the number of runtimes that can be incorporated in our technical infrastructure. Therefore, CBK models encoded in almost any programming language or format can be deployed and connected to form composites thereby extending our approach.

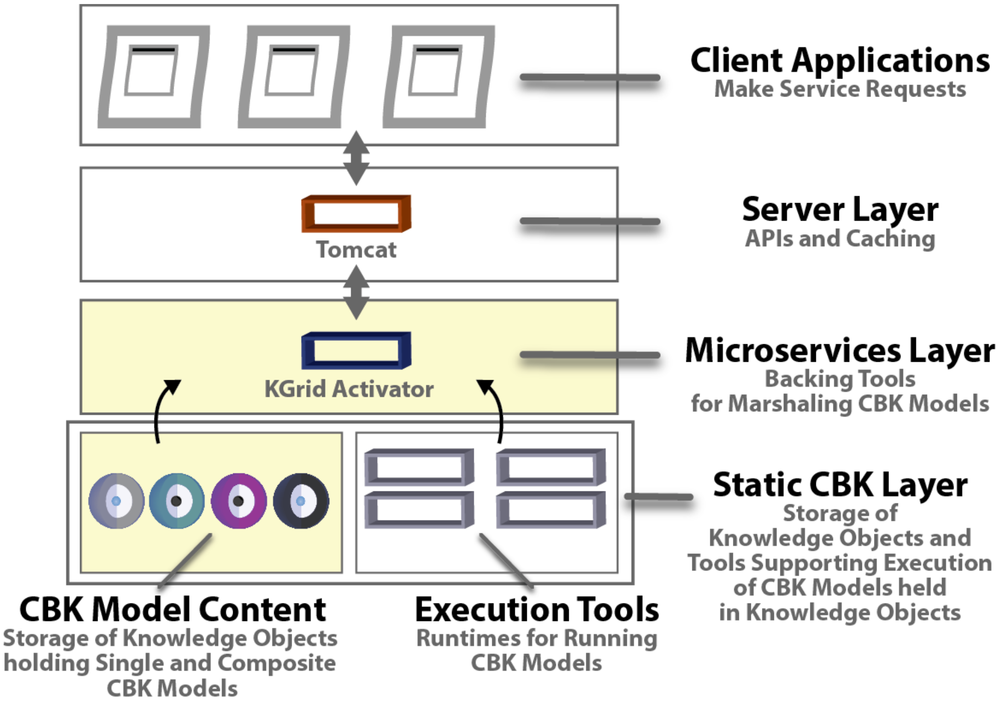


Figure 1. Technical Stack Enabling Model Composition

Next, we cover more about our approach to CBK model composition. The first key item we review in further detail is the Knowledge Object or KO packages holding CBK models (Figure 1).

KNOWLEDGE OBJECTS AND THEIR CONTENTS

Our approach to CBK model management begins with formalized Digital Objects (DOs)[28]. All DOs have three things, (1) a *bit sequence* expressing some core content, (2) *metadata* describing object properties, and (3) a *persistent unique identifier*[28]. We previously specified the new class of DOs called Knowledge Objects (KOs) diagrammed in Figure 2 below[11].
KOs provide the means to manage CBK models both as static *resources* and to deploy CBK models as *web services*. As static resources, CBK model creators and owners can place their models inside KOs as files and then transmit and share KOs over computer networks, organize KOs in digital repositories, build collections and libraries of KOs, and archive KOs for long-term safekeeping[29]. When deployed as web services, CBK models can be engaged via APIs[23,27].

We have previously shown how CBK model deployers can use KOs with the tools we have built to instantiate web services quickly and systematically[30]. The web services that result offer persistent, interactive, remote computational capabilities. These web services can also provide a mechanism to distribute CBK models directly to client applications on demand. Some of the ancillary content inside KOs exists to help establish web services for computing with CBK models.

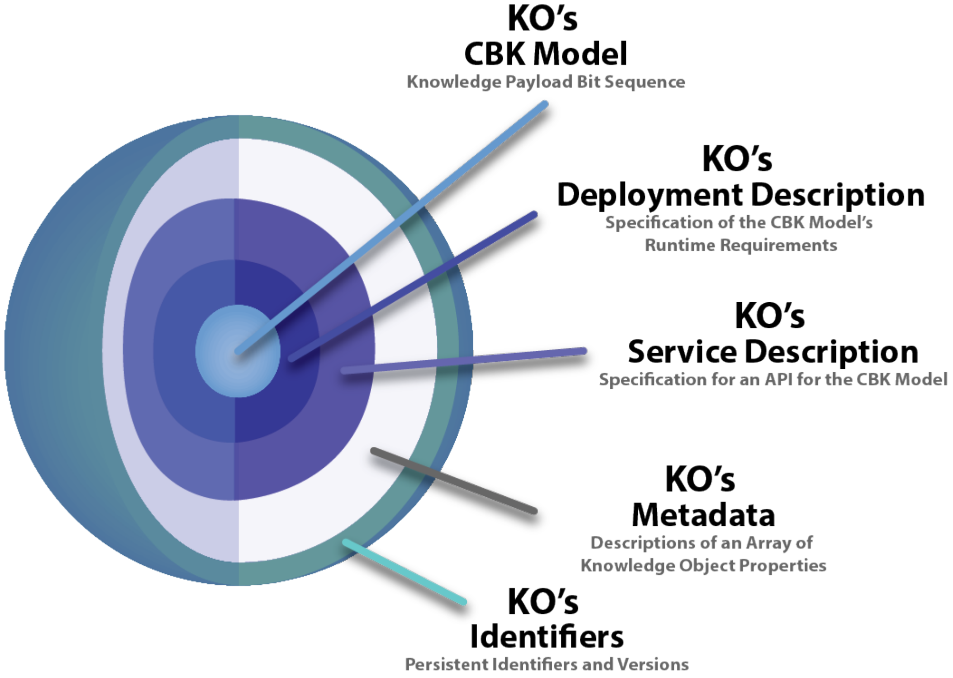

KO's
CBK Model
Knowledge Payload Bit Sequence
KO's
Deployment Description
Specification of the CBK Model's Runtime Requirements
KO's
Service Description
Specification for an API for the CBK Model
KO's
Metadata
Descriptions of an Array of Knowledge Object Properties
KO's
Identifiers
Persistent Identifiers and Versions

Figure 2. The Contents of a Knowledge Object (KO)[11], a particular subclass of Compound Digital Objects[28]

We continue exploring how best to construct easily deployable, interoperable KOs that meet the needs of multiple CBK model stakeholders, including CBK model creators, owners, organizers, deployers, and of course client application developers and users. Recently, some of our team members have closely examined the metadata needed to make large numbers of CBK models packaged in compound digital objects findable, accessible, interoperable, and reusable (the FAIR principles[31])[32]. Next, following from the innermost to outermost components shown in Figure 2 above, we give examples of the information content for each component of the generic KO.

EXAMPLE OF A SIMPLE CBK MODEL HELD INSIDE A KNOWLEDGE OBJECT

We created a KO that holds a simple CBK model (Box 1). This model uses JavaScript code to relate two variables in a common formula for body mass index (BMI). The code shown is a computer-executable representation of the mathematical function for BMI (Box 2).

```
function bmi(inputs){
  height = inputs.features.height; // height in inches
  weight = inputs.features.weight; // weight in pounds
  return weight/height/height*703;
}
```

Box 1. Body mass function encoded in the JavaScript programming language

$$BodyMassIndex\,(BMI) = \frac{\frac{bodyweight(lb)}{height(in)}}{height\,(in)} * 703$$

Box 2. Body mass index formula

The purpose of a CBK model expressing the BMI formula (Box 1) is to compute an index for body weight relative to height. Originally dubbed the Quetelet Index after its inventor and later labeled the Body Mass Index (BMI) by Keys, when concretized in code, this simple CBK model plays an ongoing role in biomedical research and practice[33]. While it makes a good example, the simplicity of this CBK model for BMI is misleading since CBK models are often much more complex.

We currently limit the CBK model content held in Knowledge Objects to explicit instances of executable code or machine-readable data and do not support Knowledge Objects containing pointers to CBK models kept elsewhere. This limitation reflects the high priority we give to making CBK models accessible and their use secure by running them nearby protected health data sources. We recognize other CBK model use cases may be better supported by using pointers.

EXAMPLE OF THE DEPLOYMENT DESCRIPTION CONTENT INSIDE A KNOWLEDGE OBJECT

Every KO carries a Deployment Description file rendered in a simple format that we devised (Box 3). Deployment Description files convey a small amount of critical content. For example, the Deployment Description given in Box 3 below specifies a suitable runtime for executing the CBK models packaged in KOs (engine:node), the name of an executable file to be used as an entry point (“bmi.js”), a list of executable artifacts (here there is only one, bmi.js), and the name of an instantiable function for computing body mass index (function: bmi).

```
/bmi:
  post:
    engine: node
    entry: bmi.js
    artifact: bmi.js
    function: bmi
```

Box 3. Example of actual Deployment Description File Content

To improve standardization, we are exploring possible conventions for representing runtime information[34]. Also, our work is so far limited to CBK models that are pure, stateless executable functions, like BMI. Pure functions associate one or more inputs to a single output[34]. Pure functions make no changes to variables outside of the function's scope, warding off software side effects. In the future, we plan to extend our work to cover stateful CBK models too.

EXAMPLE OF THE SERVICE DESCRIPTION CONTENT INSIDE A KNOWLEDGE OBJECT

Alongside CBK models, KOs hold Service Description files. These files specify an application programming interface (API) for each web service associated with a CBK model. We currently render API specifications in the machine-readable Open API 3.0 format for RESTful web service APIs[35]. Other formats could be used, such as AsyncApi 2.0 for event-driven web service APIs[36].

A snippet from an actual Service Description file in the Open API 3.0 format is provided in Box 4 below. The version of the web service version (1.0) is different from versions of the CBK model or versions of the whole KO, which appear elsewhere in our metadata. Looking at the content of the Service Description, when put together with a deployed server's IP address (not shown), the partial URL given (/ipp/bmicalculator/1.0) and the path specification (/bmi) comprise a reachable URL access API endpoint that a developer can use to engage the BMI CBK model as a webservice.

```
openapi: 3.0.0
info:
  version: '1.0'
  title: BMI Calc
  description: Calculates BMI
  license:
    name: GNU General Public License v3 (GPL-3)
    url: >-
      https://tldrlegal.com/license/gnu-general-public-license-v3-(gpl-3)#fulltext
  contact:
    name: KGrid Team
    email: kgrid-developers@umich.edu
    url: 'http://kgrid.org'
servers:
  - url: /ipp/bmicalculator/1.0
    description: BMI Calculator
tags:
  - name: BMI Calculator Endpoint
paths:
```

```
/bmi:
```

Box 4. Snippet Showing Content from an actual Service Description File

OTHER CONTENT INSIDE A KNOWLEDGE OBJECT

We have covered the CBK model, Service Description, and Deployment Description packaged inside of KOs. In addition to this content, KOs also contain a metadata file with a linked data representation of the title, authors or owners, and KO version. Inside our metadata files, one also finds the persistent unique identifier (PUID) for the KO. This PUID, along with the rest of the KO metadata, supports the search and discovery of large numbers of uniquely identified KOs.

Next, in support of our general technical approach, we describe the KGrid Activator we have built. This tool sits in the Microservices Layer of our technical stack (Figure 1).

THE KGRID ACTIVATOR MICROSERVICE TOOL

To enable our model composition method, we designed and developed the KGrid Activator. The KGrid Activator is a server-side backend tool that apps can communicate with. It is built as a Java microservice tool[37] with the help of the Java Spring Framework[38]. The KGrid Activator is KO-aware. It activates KOs and then serves as an API gateway, orchestrating the execution of the

CBK models packaged in KOs[27]. The KGrid Activator is a *reference implementation* of a backend tool. It enables us to continuously test our commitment that all models held in KOs will run.

To assist CBK model deployers, the KGrid Activator implements a repeatable pattern to "activate" CBK models. In this case, activation of CBK models is the rapid and consistent deployment of web services backed by running CBK models. To demonstrate the feasibility of our approach to technical experts who might be tasked with deploying web services backed by CBK models, we set an initial performance benchmark to "activate" CBK models held inside KOs in five seconds or less by giving simple commands to the KGrid Activator. A five second time-to-deployment is in keeping with the time required by other container-based deployment infrastructures like Docker.

Figure 3 below portrays how the KGrid Activator works to support one or more end-user client applications with web services backed by CBK models. Starting at the top of Figure 3, three client applications are shown, an EHR next to a cardiology and pathology app. These and other apps (including SMART apps and CDS Hooks) can be programmed to engage executable CBK models via calls to typical web servers (e.g., Tomcat) that in turn shuttle information to and from the web services enabled by the Activator.

Figure 3 includes four very simple example CBK models encoded in JavaScript. The first CBK model is the BMI model described previously (1.BMI). The second is a formula for body surface area (2.BSA). The third is a formula for computing cardiac output using stroke volume and heart rate (3.CO). The fourth is a formula for calculating creatinine clearance by the kidneys (4.CrCL).

In Figure 3, the first three models have been deployed and activated by an instance of the Activator, giving rise to three pairs of corresponding web services (cones) and deployed executables (dots). Once activated, the deployed executables (dots) are shown running on demand inside an instance of the JavaScript V8 runtime. Note that the fourth model (4.CrCL) is held inside a KO at the bottom of Figure 3 alongside what could potentially be many other KOs that are in storage but ready to be activated and used anytime. These stored KOs are not currently deployed.

Moving further down the technology stack depicted in Figure 3, the KGrid Activator activates KOs with the help of a specific Runtime Adapter that we have built. This adapter allows the KGrid Activator to communicate with the JavaScript V8 runtime. Using it, the Activator can command the JavaScript V8 runtime to accept and run CBK models 1, 2, and 3, as shown. Finally, on the bottom right of Figure 3, two other runtimes, Python Anaconda and GNU-R, are portrayed. These two additional runtimes have not yet been instantiated and connected to the Activator, although they could be if KOs with CBK models encoded in Python or R needed to be activated as well.

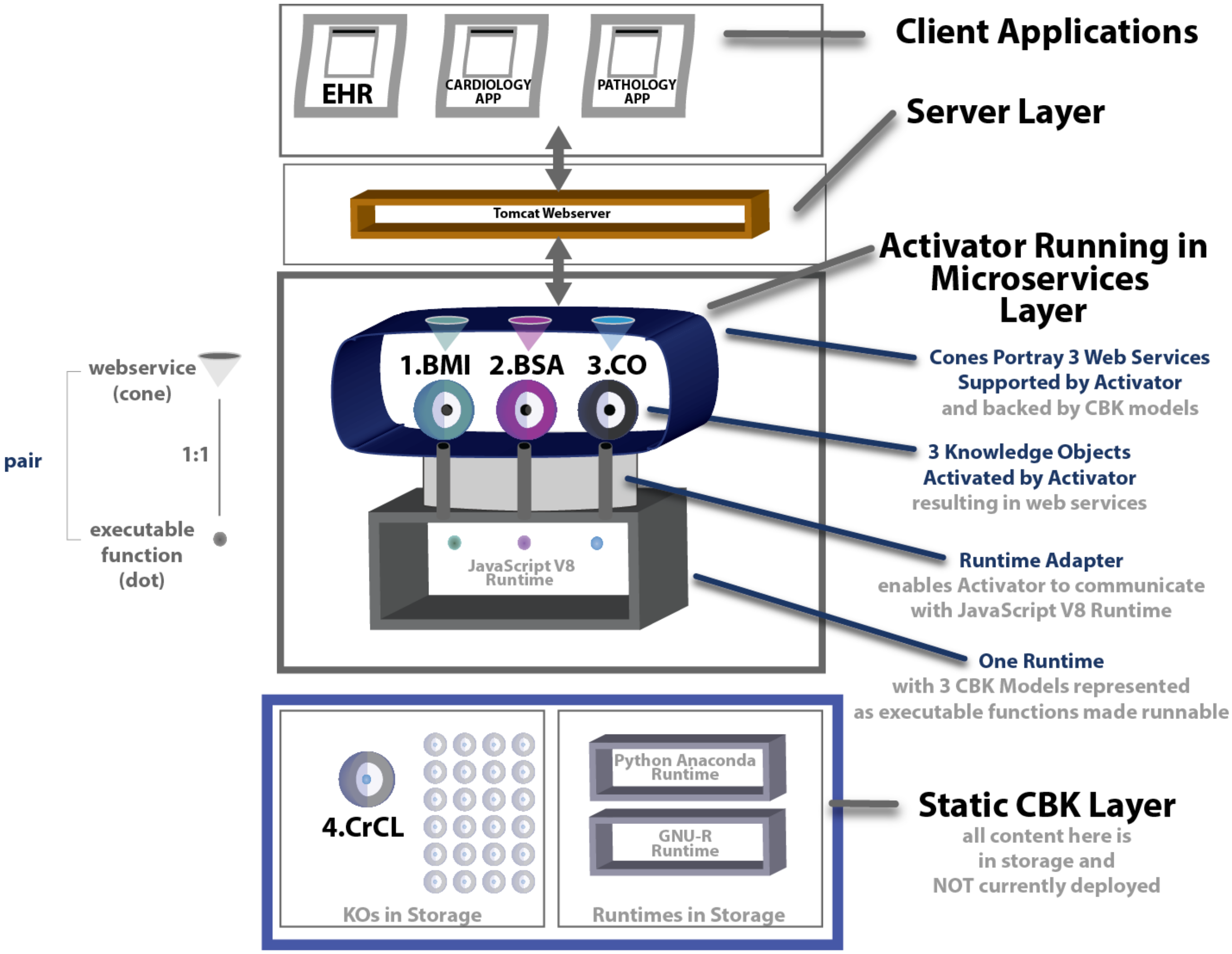

EHR
CARDIOLOGY APP
PATHOLOGY APP
Client Applications
Server Layer
Tomcat Webserver
Activator Running in Microservices Layer
webservice (cone)
pair
1:1
executable function (dot)
1.BMI
2.BSA
3.CO
JavaScript V8 Runtime
Cones Portray 3 Web Services Supported by Activator
and backed by CBK models
3 Knowledge Objects Activated by Activator
resulting in web services
Runtime Adapter
enables Activator to communicate with JavaScript V8 Runtime
One Runtime
with 3 CBK Models represented as executable functions made runnable
4.CrCL
KOs in Storage
Python Anaconda Runtime
GNU-R Runtime
Runtimes in Storage
Static CBK Layer
all content here is in storage and NOT currently deployed

Figure 3. View of the KGrid Activator operating in the Microservices Layer

Summarizing, in Figure 3, activation of three CBK models for body mass index (BMI), body surface area (BSA), and cardiac output (CO) establishes three corresponding web services (green, pink, and blue cones). After the KGrid Activator provides web service endpoints to the Server Layer, it plays an API gateway role by routing incoming requests from external Client Applications to the CBK models running as executable functions inside one or more deployed runtimes. Now that we have covered the content of KOs, activation and the KGrid Activator, the last item to introduce as part of our technical methods is the *Runtime Context*.

HOW A RUNTIME CONTEXT ENABLES CBK MODEL COMPOSITION

As part of the work of activating CBK models held inside KOs, each instance of the KGrid Activator establishes and maintains a dynamic list of every CBK model it has deployed with an internal reference pointer to each deployed CBK model (Figure 4). This list is the *Runtime Context*.

The Runtime Context provided by the KGrid Activator is accessible programmatically. For our method of model composition, when a request from the code of one CBK model is made to use another CBK model the KGrid Activator uses its current Runtime Context to operate as a gateway. In its gateway role, the KGrid Activator resolves CBK model references and oversees request-response execution. This is how one deployed CBK models can call on any other deployed CBK model without having to know its implementation or instantiation details (Figure 4).

Going deeper into how this works, complete resolution of the reference pointers to deployed CBK models is achieved by the KGrid Activator working in conjunction with each runtime. This is where our runtime adapters come into play. We provide adapters to handle programming language and runtime-specific concerns. (As one example, an adapter for the JavaScript V8 runtime is depicted in Figure 3.) Each adapter defines and implements a common communication protocol (i.e., interface) we developed to integrate different runtimes with the KGrid Activator. Upon successful deployment of a CBK model into its corresponding runtime by the KGrid Activator, the model is registered by the adapter resulting in a new entry to the KGrid Activator's current Runtime Context. As a result, working in conjunction with the KGrid Activator, adapters allow CBK model-to-model request-response activity to finally happen.

The Runtime Context is critical for interconnecting models and enabling model composition. For now, we have this capability only when every CBK model being connected is deployed inside the same runtime. We see a need for flexibility and power and plan to develop new capabilities so that future model composition can span multiple runtimes with the help of an enhanced KGrid Activator. Once additional CBK model-to-model request-response capabilities are developed, KOs handling separate parts of the complex calculations can be deployed in separate runtimes. This will make it possible to build composite CBK models using component CBK models that are encoded in more than one format or programming language.

To summarize, as portrayed in Figure 4 below, the Runtime Context is a dynamic list or map maintained by an instance of the KGrid Activator. The Runtime Context supports all possible request-response calls between activated CBK models running inside one runtime instance at any moment in time. As the number of CBK models deployed into that runtime instance changes, the Runtime Context is consistently refreshed and remains current. This approach is what finally enables our technology stack to support CBK model composition.

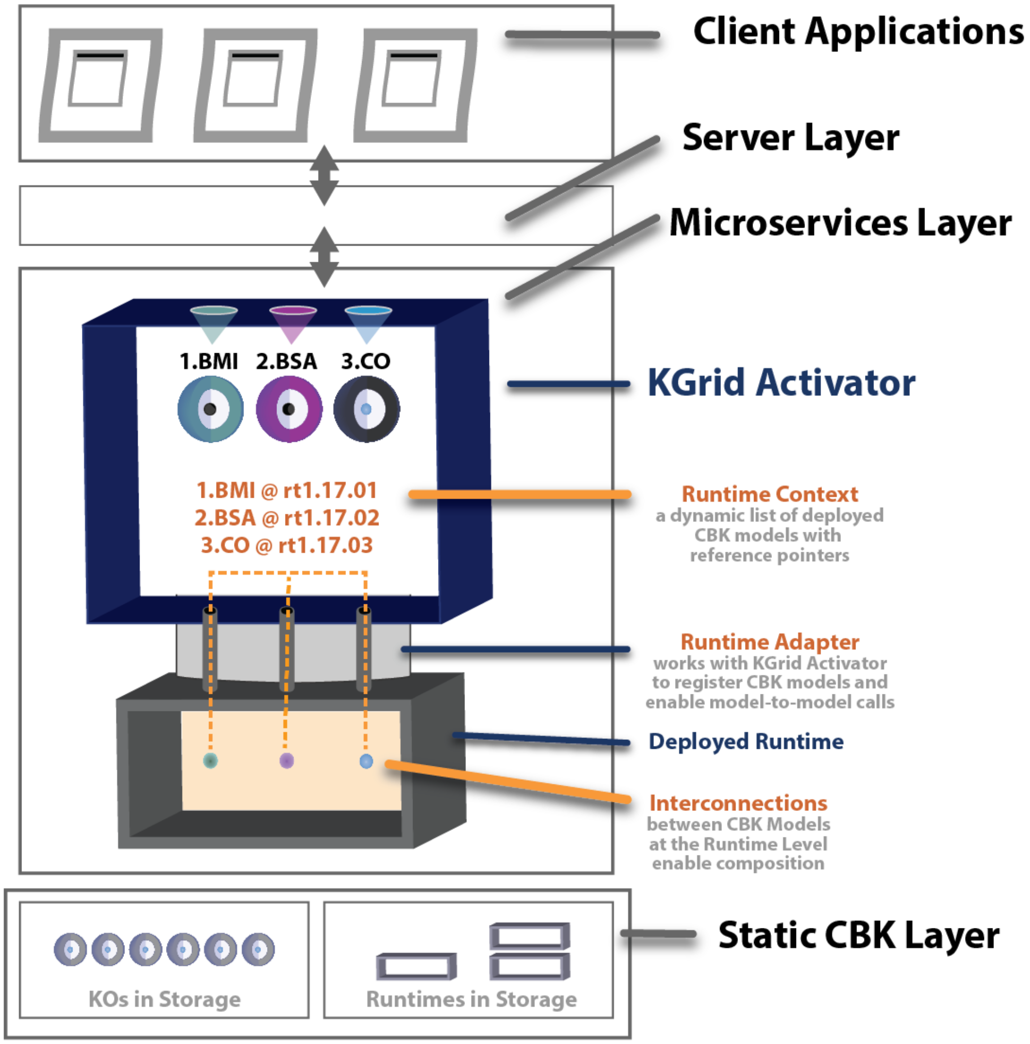


Figure 4. Inside the KGrid Activator, the Runtime Context is a dynamic list of CBK Models running locally

CREATING COMPOSITE CBK MODELS

Instead of building all new technology for doing model composition, the KGrid Activator marshals the specific contents of KOs and then uses existing web server and runtime technologies for model composition. Together, the modular components in our technology stack are sufficient to create three general kinds of composite models (Figure 5). These three kinds of composite models – serial, hierarchical, and conditional – derive from a larger collection of common computational workflow patterns that have appeared repeatedly over the years in many other computer science works[39].

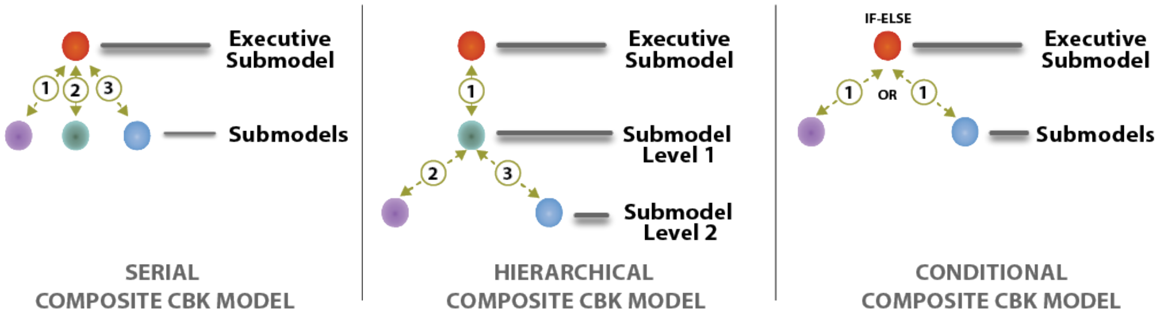


Figure 5. Three General Kinds of Composite CBK Models

On the left in Figure 5, a *serial composite CBK model* relies on the procedural knowledge about a model composition represented in an Executive Submodel (orange dot) to engage three submodels sequentially in steps 1, 2, and 3.

Similarly, in the middle of Figure 5, a *hierarchical composite CBK model* has a top-level Executive Submodel connected to another Executive Submodel at Level 1 (green dot), which in turn encodes procedures to connect the two other submodels at Level 2 (purple and blue dots).

Additionally, it is possible to insert conditional "IF-THEN-ELSE" logic into composite models. As an example of this, on the right in Figure 5 is a *conditional composite CBK model* with an Executive Submodel that, depending on some condition, connects to one or the other (but not both) of the two submodels shown (purple and blue dots). The procedural knowledge encoded into the logic of Executive Submodels expresses model compositions. All three general kinds of composite models in Figure 5 – and mixtures of them – can be composed using our technical approach.

TECHNICAL FEASIBILITY DEMONSTRATION: BUILDING AND USING THE CM-IPP

As a demonstration of the feasibility of our technical approach to model composition, we created the *Composite Model for Individualized Precision Prevention* (CM-IPP). We used the CM-IPP to conduct a concordance study which we will report elsewhere. (The concordance study sought to answer the research question. "To what degree do primary care providers collectively agree with the value-based rankings of 21 preventive services computed by the CM-IPP?")

According to the United States Preventive Services Task Force (USPSTF), the 21 preventive services covered by the CM-IPP are supported by high-quality "A" or "B" level evidence. Examples of these services include screening services (e.g., screening for colorectal cancer) and other measures (e.g., taking aspirin to lower the risk of a heart attack). Studies show that in the U.S., many people receive some recommended preventive services, but few receive them all[26]. The ultimate purpose of composite models like the CM-IPP is to enable computerized individualized prioritization of recommended preventive services during routine primary care encounters. Next, for our results, we provide a detailed technical description of the CM-IPP.

RESULTS:

We successfully re-implemented a multi-component statistical model created and originally implemented as a Markov Cohort model inside a spreadsheet by Taksler and colleagues as part of their work in preventive medicine[26]. Working with Dr. Taksler and his group, using KOs and our technical approach, we arrived at the new CM-IPP composite CBK model for computing an individual's life-gain arising for 21 recommended preventive medical services. When client applications engage the new CM-IPP, the applications receive a ranked list of relevant preventive services for an individual. The CM-IPP applies static and simplified results of Markov Cohort modeling to compute estimated life-gain from regression formulas where some Markov Cohort model information is lost. In the future, the Taksler research team plans to use microsimulation modeling techniques[40] and provide results either as lookup tables or mathematical equations.

CM-IPP INPUTS AND OUTPUTS

The input to the CM-IPP includes more than 100 features about a person. Any client application with these input data can post an instance of our proprietary JSON data object (Box 5) to the web service backed by the CM-IPP's top-level Executive Submodel. Client applications receive computed rankings of 21 relevant preventive services by estimated lifespan gain in years (Box 6).

```
INPUT {
    "patient":{
      "id": "ipp-patient-E01",
      "features":{
        "age":50,
        "race":"Black",
        "gender":"Male",
        "height":73.0,
        "weight":220.0,
        "systolic":147,
        "diastolic":68,
        "totalcholesterol":110,
        "HDL":32,
        "LDL":42,
        "triglycerides":181,
        "a1c":9,
        "cvd":false,
          [ 88 more features listed as key:value pairs go here ]
```

Box 5. Partial List of Inputs for the CM-IPP web service

```
OUTPUT "lifeexpectancy": {
      "aspirinPrevention": {
        "total": {
          "life-gain": 0.3175741769690106 }
      },
      "crcScreening": {
        "total": {
          "life-gain": 0.0919283879684265 }
      },
      "diabetesControl": {
        "total": {
          "life-gain": 1.5806718157369808 }
     },
     [ more computed results go here ]
```

Box 6. Partial List of Outputs Provided by the CM-IPP web service

MORE *TECHNICAL DETAILS ABOUT THE CM-IPP*

The CM-IPP is a mixed *serial-hierarchical-conditional* composite model that performs a long and complex call chain of computations about an individual. This complex call chain (not shown) takes advantage of the 42 submodels of the CM-IPP that are arrayed graphically in Figure 6 below.

As depicted in Figure 6, the CM-IPP's top-level Executive Submodel calls on 29 other submodels directly, three of which are lower-level Executive Submodels (orange dots). The three lower-level Executive Submodels call on a total of 13 more submodels. Conditional logic inside the top-level Executive Submodel makes it so that only the relevant Net Benefit submodels (gray dots) are engaged while doing computations about a given person. In addition, the top-level Executive Submodel includes more computable knowledge to compute benefit estimates for five preventive services directly.

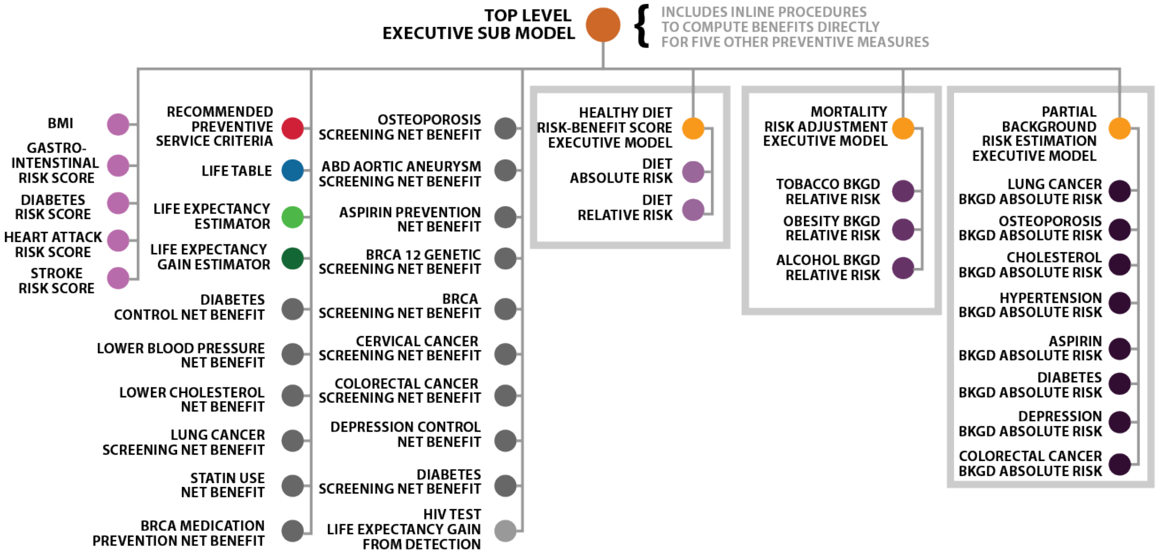

TOP LEVEL
EXECUTIVE SUB MODEL
INCLUDES INLINE PROCEDURES
TO COMPUTE BENEFITS DIRECTLY
FOR FIVE OTHER PREVENTIVE MEASURES
BMI
GASTRO-INTESTINAL RISK SCORE
DIABETES RISK SCORE
HEART ATTACK RISK SCORE
STROKE RISK SCORE
RECOMMENDED PREVENTIVE SERVICE CRITERIA
LIFE TABLE
LIFE EXPECTANCY ESTIMATOR
LIFE EXPECTANCY GAIN ESTIMATOR
DIABETES CONTROL NET BENEFIT
LOWER BLOOD PRESSURE NET BENEFIT
LOWER CHOLESTEROL NET BENEFIT
LUNG CANCER SCREENING NET BENEFIT
STATIN USE NET BENEFIT
BRCA MEDICATION PREVENTION NET BENEFIT
OSTEOPOROSIS SCREENING NET BENEFIT
ABD AORTIC ANEURYSM SCREENING NET BENEFIT
ASPIRIN PREVENTION NET BENEFIT
BRCA 12 GENETIC SCREENING NET BENEFIT
BRCA SCREENING NET BENEFIT
CERVICAL CANCER SCREENING NET BENEFIT
COLORECTAL CANCER SCREENING NET BENEFIT
DEPRESSION CONTROL NET BENEFIT
DIABETES SCREENING NET BENEFIT
HIV TEST LIFE EXPECTANCY GAIN FROM DETECTION
HEALTHY DIET RISK-BENEFIT SCORE EXECUTIVE MODEL
DIET ABSOLUTE RISK
DIET RELATIVE RISK
MORTALITY RISK ADJUSTMENT EXECUTIVE MODEL
TOBACCO BKGD RELATIVE RISK
OBESITY BKGD RELATIVE RISK
ALCOHOL BKGD RELATIVE RISK
PARTIAL BACKGROUND RISK ESTIMATION EXECUTIVE MODEL
LUNG CANCER BKGD ABSOLUTE RISK
OSTEOPOROSIS BKGD ABSOLUTE RISK
CHOLESTEROL BKGD ABSOLUTE RISK
HYPERTENSION BKGD ABSOLUTE RISK
ASPIRIN BKGD ABSOLUTE RISK
DIABETES BKGD ABSOLUTE RISK
DEPRESSION BKGD ABSOLUTE RISK
COLORECTAL CANCER BKGD ABSOLUTE RISK

Figure 6. Graphical View of the CBK Composite Model for Individualized Precision Prevention (CM-IPP)

Overall, we used 11 different types of submodels to build the CM-IPP. Table 1 has descriptions of each of the 11 submodel types that contribute to the overall CM-IPP.

Submodel Type	Description of Submodel Type	Quantity in CBK-CM-IPP
Executive	These include one top-level and three lower-level mostly procedural submodels with logic representing model composites. The web service backed by the top-level Executive model accepts a large data object as its input from an external Client Application to kickoff CM-IPP processing. The three lower-level executive submodels compute various intermediate results with the help of other submodels.	4
Patient Feature Derivation	These submodels further process original patient data input to derive additional patient features (i.e., BMI and common risk scores).	5
Recommended Preventive Service Criteria	This submodel encodes inclusion and exclusion criteria, including age ranges, sex, and other criteria for all included preventive medical services covered in the CM-IPP.	1
Life Table	This submodel contains final numeric results for baseline mortality rates by age, race, and sex from the US Centers for Disease Control and Prevention.	1
Life Expectancy Estimator	This submodel calculates an individual’s life expectancy using a given age range and a previously computed set of age-ordered mortality rates.	1
Life Expectancy Gain Estimator	This submodel accepts two life expectancy estimates computed with the Life Expectancy Estimator submodel as inputs and computes their difference.	1
Net Benefit Estimators	These submodels compute the estimated net benefit in terms of marginal life-gain arising from implementing most preventive services and related interventions.	15
Detection Gain Estimator	This submodel calculates an individual’s life expectancy gain from detection of HIV through routine testing.	1
Diet Risk Models	These two submodels estimate relative and absolute risks related to maintaining or not maintaining a healthy diet.	2
Mortality Risk Adjustment	These three submodels support computed adjustments in all-cause mortality risk based on an individual’s degree of obesity and alcohol and tobacco use.	3
Partial Background Risk Estimation	These submodels compute background absolute risk for several diseases (e.g., lung cancer) and several treatments or interventions (e.g., taking aspirin.)	8

Table 1. The Eleven Types of Submodels connected to create the CBK Composite Model for IPP (CM-IPP)

USING THE CM-IPP IN PREVENTIVE SERVICES RESEARCH

Table 2 summarizes the end results of computations made using the CM-IPP for 12 fictitious but realistic patient cases. These cases are part of a research study about preventive services.

Fictitious Patient Case	Source of Ranking	Top-ranked Service	Second-ranked Service	Third-ranked Service	**Fictitious Patient Case**	Source of Ranking	Top-ranked Service	Second-ranked Service	Third-ranked Service
#4	CM-IPP	BRE	CRC	DIET	**#2**	CM-IPP	LUN	STA	DIET
#6	CM-IPP	BP	SMO	ALC	**#7**	CM-IPP	CRC	DIET	AAA
#5	CM-IPP	SMO	BP	DIET	**#8**	CM-IPP	DIET	WEI	LUN
#11	CM-IPP	SMO	DIET	BP	**#10**	CM-IPP	BRE	CRC	DIET
#1	CM-IPP	DIET	CRC	ASA	**#12**	CM-IPP	DIET	ALC	SMO
#3	CM-IPP	ALC	DIA	BP	**#9**	CM-IPP	ALC	DIET	WEI

Table 2: For each fictitious case, the Top 3 ranked services computed by the CM-IPP are listed. The letter codes stand for abdominal aortic aneurysm screening (AAA), reducing alcohol use (ALC), aspirin use (ASA), treating blood pressure (BP), breast cancer screening (BRE), colorectal cancer screening (CRC), treating diabetes (DIA), diet counseling (DIET), statin use for cholesterol (STA), smoking cessation (SMO), and losing 10 pounds of weight (WEI).

DISCUSSION:

We set out to compose distinct biomedical models using a repeatable and transferable method and achieved our goal. We built the CM-IPP, a composite model that reasons and computes with 42 submodels representing biomedical knowledge from different sources (e.g., AHRQ and CDC).

Our model composition method makes a strong commitment to decentralized web technology. For client applications that can access the web, our method takes advantage of HTTP server technology, common web service architectural patterns, and publicly available runtimes, such as JavaScript V8. Thus, our work aligns well with that of the WWW community and its massive installed base.

On the plus side, much of the software in this installed base interoperates and scales well already. However, we also inherit many challenges with the WWW. First among them, to make good on our approach, we must gain widespread adoption of new model packaging and activation specifications implemented using Knowledge Objects and the KGrid Activator, respectively. Second, all WWW security issues apply. At every level of our technology stack, the implementation of security protocols is required to protect CBK model stakeholders from harm. Third, many issues remain with measuring and expressing the quality of CBK models. We have demonstrated how to compose CBK models but not how to ensure the models composed are safe and effective for their intended use. In our future work, we anticipate using more than one evidence-based composite model to generate advice and then using other composite models to

prioritize competing and conflicting advice based on evidence grades and other relevant factors.
In this initial work we have achieved a measure of *syntactic interoperability* via RESTful APIs but have yet to work on *semantic interoperability* for the inputs and outputs of CBK models[41]. We have also packaged the CM-IPP so that client application developers can build it directly into a code base instead of engaging it as a web service.

For composite model design, our work highlights the need for more and better principles to guide submodel modularization and composition. We struggled with issues of submodel scope and performance. For example, the top-level Executive Submodel in the CM-IPP mixes overarching procedural code for model composition with domain-specific logic for five preventive services, making it more difficult to reuse this submodel in other contexts. Also, in hindsight, the 15 submodels used for net benefit estimation might have built into a single submodel instead.

Our experience does at least suggest two principles to inform decisions about submodel scope. First, a distinct submodel may be justified when it is subject to very frequent updates. Second, a distinct submodel may be justified if it can be reused in many composite models. Otherwise, the extra work of packaging and managing a distinct submodel seems not to be worth it.

Our work on model composition raises important knowledge management issues too. One of them is versioning many items, including the parts of KOs and model subcomponents, composite models, and the web services backed by such models. Good dependency management relies on item-level versioning. We wish to extend our version-control capabilities by applying lessons learned by those who have used software package metadata, automation, and human workflows to keep up with direct and transitive dependencies in decentralized web systems.

Another knowledge management challenge is how to categorize different types of submodels. We labeled any submodel containing a part or whole representation of a composite model as an "Executive Submodel." In addition, we attempted to categorize submodels further by their function and the outputs they produce. We found this somewhat difficult to do and think our current ad hoc submodel categories are shaky and that better methods of submodel categorization are needed.

CONCLUSION

In support of biomedical research and learning health systems, we offer a new method for CBK model composition that relies on decentralized web technology. Our method uses special digital objects called Knowledge Objects to establish operable pairs of web services and running computer-processable functions. In this way, complex composite models are made accessible via the World Wide Web. As a feasibility demonstration, we connected 42 packaged submodels into a composite CBK model called CM-IPP. We then used the CM-IPP to compute individualized life-gain estimates for a host of evidence-based preventive services. By accessing CM-IPP via a web service, client applications gain powerful computational and reasoning capabilities quickly and easily. Using this approach, CBK model-makers can build network-accessible composite models to advance research, education, and learning health systems.

PROJECT SPONSOR

The Agency Healthcare Research and Quality sponsored this work through R21 grant number HS026257-01. More information about this funded project is available here.

ACKNOWLEDGMENTS

We are very grateful for the expert professional support of Astrid Fishstrom and Lisa Ferguson in the Department of Learning Health Sciences at the University of Michigan Medical School. For their insights about this work, we thank our colleagues, Zach Landis-Lewis, Rachel Richesson, Josh Rubin, and Doug Van Houweling.

CONFLICTS OF INTEREST

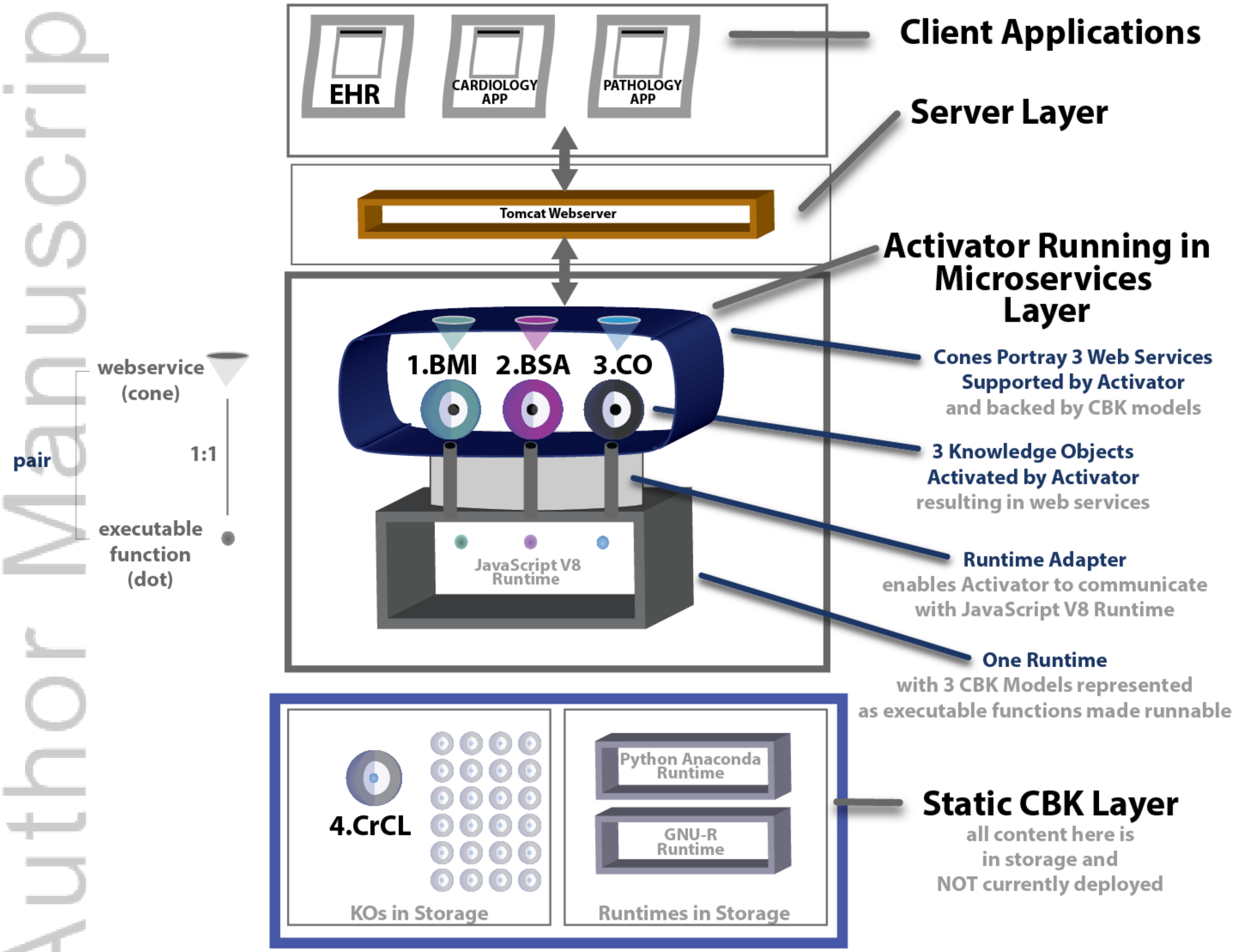
Dr. Taksler received consulting fees from the University of Michigan, Ann Arbor on a grant funded by the Agency for Healthcare Research and Quality (R21HS026257), directly related to the submitted work. The other authors assert they have no conflicts of interest.

REFERENCES

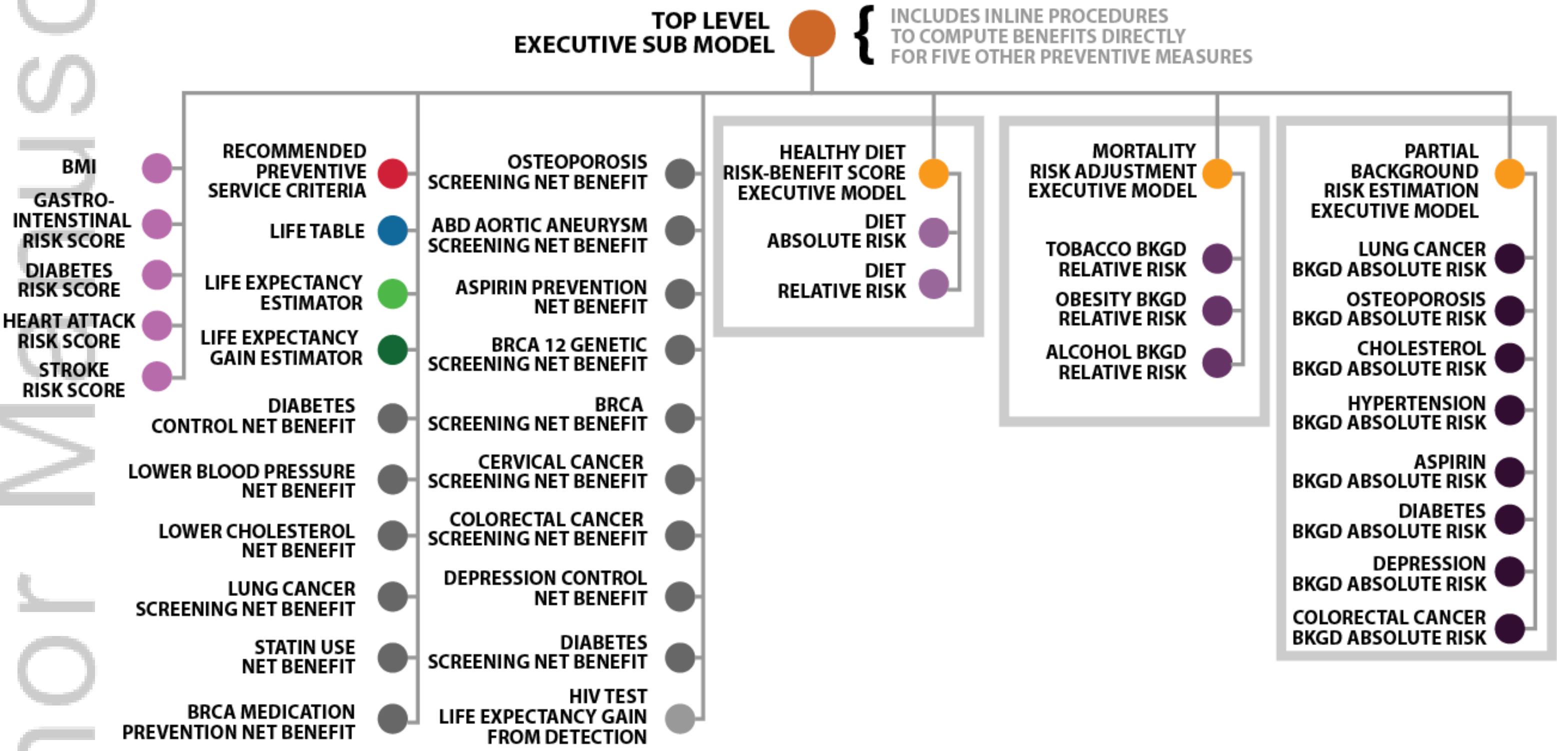
1. Tsafnat G, Coiera EW. Computational reasoning across multiple models. Journal of the American Medical Informatics Association. 2009 Nov 1;16(6):768-74.
2. Blaom AD, Kiraly F, Lienart T, Simillides Y, Arenas D, Vollmer SJ. MLJ: A Julia package for composable machine learning. arXiv preprint arXiv:2007.12285. 2020 Jul 23.
3. Allen R, Garlan D. A formal basis for architectural connection. ACM Transactions on Software Engineering and Methodology (TOSEM). 1997 Jul 1;6(3):213-49.
4. Bézivin J, Bouzitouna S, Del Fabro MD, Gervais MP, Jouault F, Kolovos D, Kurtev I, Paige RF. A canonical scheme for model composition. In European Conference on Model Driven Architecture-Foundations and Applications 2006 Jul 10 (pp. 346-360). Springer, Berlin, Heidelberg.
5. Tang K, Liu X, Harper SL, Steevens JA, Xu R. NEIMiner: nanomaterial environmental impact data miner. International journal of nanomedicine. 2013;8(Suppl 1):15.
6. Purcell O, Jain B, Karr JR, Covert MW, Lu TK. Towards a whole-cell modeling approach for synthetic biology. Chaos: An Interdisciplinary Journal of Nonlinear Science. 2013 Jun 13;23(2):025112.
7. Mulder S, Hamidi H, Kretzler M, Ju W. An integrative systems biology approach for precision medicine in diabetic kidney disease. Diabetes, Obesity and Metabolism. 2018 Oct;20:6-13.
8. Friedman CP, Flynn AJ. Computable knowledge: an imperative for learning health systems. Learning health systems. 2019 Oct;3(4).
9. Zeigler BP, Mittal S, Traore MK. Fundamental requirements and DEVS approach for modeling and simulation of complex adaptive system of systems: Healthcare reform. In Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems 2018 Apr 15 (pp. 1-12).
10. Seidewitz E. What models mean. IEEE software. 2003 Sep 15;20(5):26-32.
11. Flynn AJ, Friedman CP, Boisvert P, Landis-Lewis Z, Lagoze C. The Knowledge Object Reference Ontology (KORO): a formalism to support management and sharing of computable biomedical knowledge for learning health systems. Learning Health Systems. 2018 Apr;2(2):e10054.
12. Lomotan EA, Meadows G, Michaels M, Michel JJ, Miller K. To share is human! Advancing evidence into practice through a national repository of interoperable clinical decision support. Applied clinical informatics. 2020 Jan;11(01):112-21.
13. De Smedt K, Koureas D, Wittenburg P. FAIR Digital Objects for Science: From Data Pieces to Actionable Knowledge Units. Publications. 2020 Jun;8(2):21.
14. Drexl, Hilty et al., Technical Aspects of Artificial Intelligence: An Understanding from an Intellectual Property Law Perspective, Version 1.0, October 2019, available at: https://ssrn.com/abstract=3465577
15. Müller ME. Relational Knowledge Discovery. Cambridge University Press; 2012 Jun 21.
16. Floridi L. Semantic information and the network theory of account. Synthese. 2012 Feb 1;184(3):431-54.
17. Speer R, Havasi C. ConceptNet 5: A large semantic network for relational knowledge. In The People’s Web Meets NLP 2013 (pp. 161-176). Springer, Berlin, Heidelberg.

18. Morris ZS, Wooding S, Grant J. The answer is 17 years, what is the question: understanding time lags in translational research. Journal of the Royal Society of Medicine. 2011 Dec;104(12):510-20.
19. Alavi M, Leidner DE. Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS quarterly. 2001 Mar 1:107-36.
20. Krishnamurthi S, Fisler K. Programming paradigms and beyond. The Cambridge Handbook of Computing Education Research. 2019;37.
21. Harrison R, Samaraweera LG, Dobie MR, Lewis PH. Comparing programming paradigms: an evaluation of functional and object-oriented programs. Software Engineering Journal. 1996 Jul 1;11(4):247-54.
22. Neal ML, Cooling MT, Smith LP, Thompson CT, Sauro HM, Carlson BE, Cook DL, Gennari JH. A reappraisal of how to build modular, reusable models of biological systems. PLoS Comput Biol. 2014 Oct 2;10(10):e1003849.
23. De Meester B, Seymoens T, Dimou A, Verborgh R. Implementation-independent function reuse. Future Generation Computer Systems. 2020 Sep 1;110:946-59
24. Goswami P, Gupta S, Li Z, Meng N, Yao D. Investigating The Reproducibility of NPM Packages. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) 2020 Sep 1 (pp. 677-681). IEEE.
25. Niephaus F, Felgentreff T, Hirschfeld R. Towards polyglot adapters for the graalvm. In Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming 2019 Apr 1 (pp. 1-3).
26. Taksler GB, Keshner M, Fagerlin A, Hajizadeh N, Braithwaite RS. Personalized estimates of benefit from preventive care guidelines: a proof of concept. Annals of internal medicine. 2013 Aug 6;159(3):161-8.
27. Flynn AJ, Boisvert P, Lagoze C, Meng G, Friedman CP. Architecture and initial development of a knowledge-as-a-service activator for computable knowledge objects for health. Building Continents of Knowledge in Oceans of Data: The Future of Co-Created eHealth. 2018 May 18;247:401.
28. Wittenburg P, Strawn G, Mons B, Boninho L, Schultes E. Digital objects as drivers towards convergence in data infrastructures. Technical paper. doi. 2018;10:b2share.
29. Flynn AJ, Bahulekar N, Boisvert P, Lagoze C, Meng G, Rampton J, Friedman CP. Architecture and initial development of a digital library platform for computable knowledge objects for health. In Informatics for Health: Connected Citizen-Led Wellness and Population Health 2017 (pp. 496-500). IOS Press.
30. *In Press*. Flynn AJ, Huang C, Lampa N, Meng, G, Gittlen N, Beck A, Raths B, Boisvert P. An Experiment to Convert Structured Product Labels into Computable Prescribing Information. In Proceedings, 9th IEEE International Conference on Healthcare Informatics, August 2021.
31. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, Blomberg N, Boiten JW, da Silva Santos LB, Bourne PE, Bouwman J. The FAIR Guiding Principles for scientific data management and stewardship. Scientific data. 2016 Mar 15;3(1):1-9.
32. Alper BS, Flynn A, Bray BE, Conte ML, Eldredge C, Gold S, Greenes RA, Haug P, Jacoby K, Koru G, McClay J. Categorizing metadata to help mobilize computable biomedical knowledge. 2021.

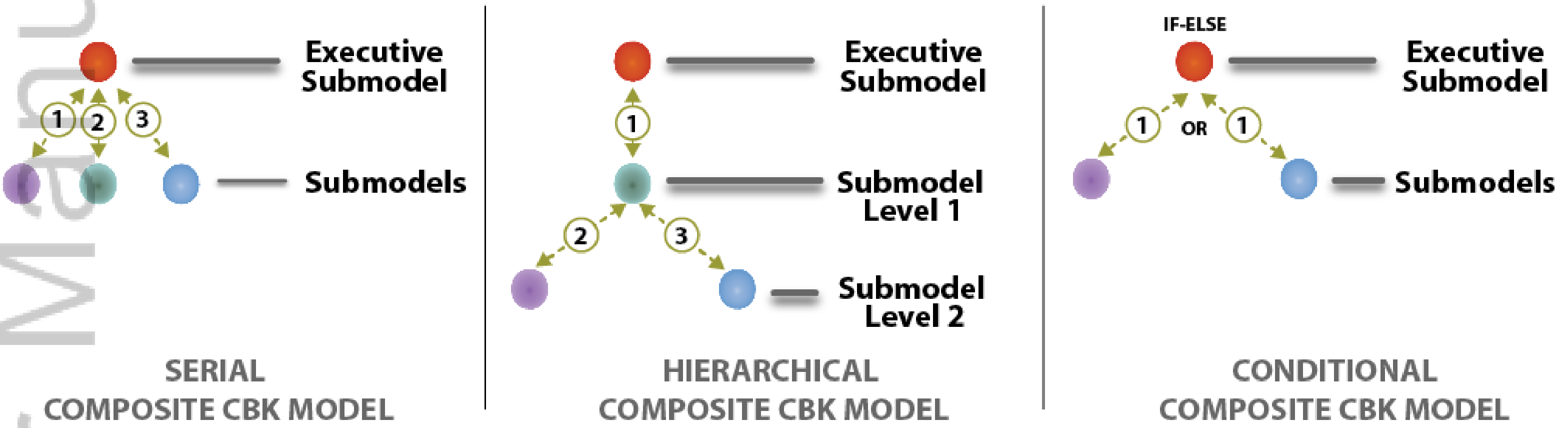
33. Garabed E, Adolphe Quetelet (1796–1874)—the average man and indices of obesity, Nephrology Dialysis Transplantation, Volume 23, Issue 1, January 2008, Pages 47–51, https://doi.org/10.1093/ndt/gfm517
34. De Meester B, Dimou A, Verborgh R, Mannens E. An ontology to semantically declare and describe functions. In European Semantic Web Conference 2016 May 29 (pp. 46-49). Springer, Cham.
35. Fielding RT. REST: architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California. 2000.
36. Gómez A, Iglesias-Urkia M, Urbieta A, Cabot J. A model-based approach for developing event-driven architectures with AsyncAPI. InProceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems 2020 Oct 18 (pp. 121-131).
37. Microservices. https://spring.io/microservices Accessed April 10, 2022.
38. Spring. https://spring.io/ Accessed April 10, 2022.
39. Knyazkov KV, Kovalchuk SV, Tchurov TN, Maryin SV, Boukhanovsky AV. CLAVIRE: e-Science infrastructure for data-driven computing. Journal of Computational Science. 2012 Nov 1;3(6):504-10.
40. Treeage. https://www.treeage.com/ Accessed April 10, 2022.
41. Serrano D, Stroulia E, Lau D, Ng T. Linked REST APIs: a middleware for semantic REST API integration. In 2017 IEEE International Conference on Web Services (ICWS) 2017 Jun 25 (pp. 138-145). IEEE.


EHR
CARDIOLOGY APP
PATHOLOGY APP
Client Applications
Server Layer
Tomcat Webserver
Activator Running in Microservices Layer
webservice (cone)
pair
1:1
executable function (dot)
1.BMI
2.BSA
3.CO
JavaScript V8 Runtime
Cones Portray 3 Web Services Supported by Activator
and backed by CBK models
3 Knowledge Objects Activated by Activator
resulting in web services
Runtime Adapter
enables Activator to communicate with JavaScript V8 Runtime
One Runtime
with 3 CBK Models represented as executable functions made runnable
4.CrCL
KOs in Storage
Python Anaconda Runtime
GNU-R Runtime
Runtimes in Storage
Static CBK Layer
all content here is in storage and NOT currently deployed

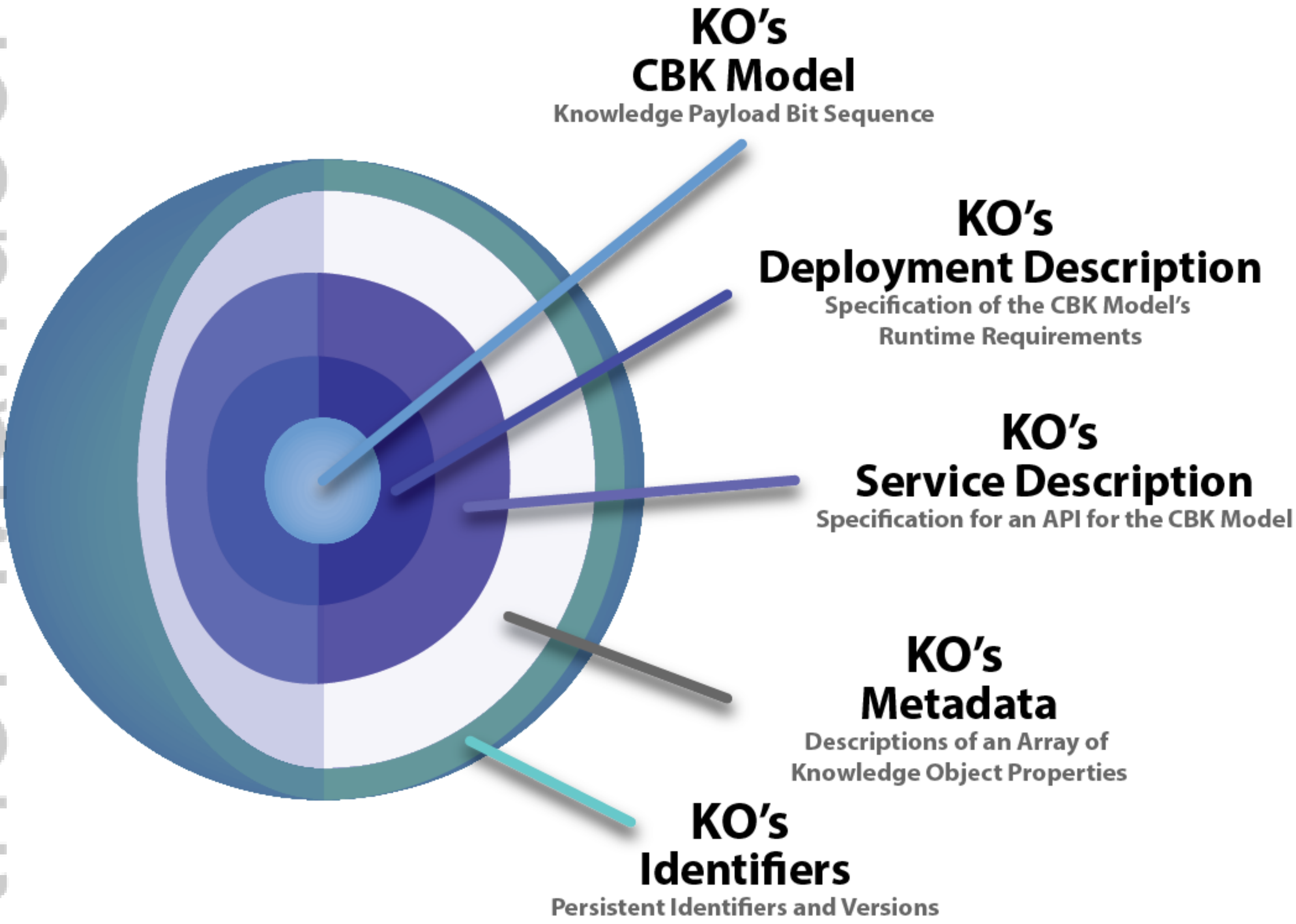
LRH2_10325_ActivatorGraphic2021.v7.png


TOP LEVEL EXECUTIVE SUB MODEL
INCLUDES INLINE PROCEDURES TO COMPUTE BENEFITS DIRECTLY FOR FIVE OTHER PREVENTIVE MEASURES
BMI
GASTRO-INTENSTINAL RISK SCORE
DIABETES RISK SCORE
HEART ATTACK RISK SCORE
STROKE RISK SCORE
RECOMMENDED PREVENTIVE SERVICE CRITERIA
LIFE TABLE
LIFE EXPECTANCY ESTIMATOR
LIFE EXPECTANCY GAIN ESTIMATOR
DIABETES CONTROL NET BENEFIT
LOWER BLOOD PRESSURE NET BENEFIT
LOWER CHOLESTEROL NET BENEFIT
LUNG CANCER SCREENING NET BENEFIT
STATIN USE NET BENEFIT
BRCA MEDICATION PREVENTION NET BENEFIT
OSTEOPOROSIS SCREENING NET BENEFIT
ABD AORTIC ANEURYSM SCREENING NET BENEFIT
ASPIRIN PREVENTION NET BENEFIT
BRCA 12 GENETIC SCREENING NET BENEFIT
BRCA SCREENING NET BENEFIT
CERVICAL CANCER SCREENING NET BENEFIT
COLORECTAL CANCER SCREENING NET BENEFIT
DEPRESSION CONTROL NET BENEFIT
DIABETES SCREENING NET BENEFIT
HIV TEST LIFE EXPECTANCY GAIN FROM DETECTION
HEALTHY DIET RISK-BENEFIT SCORE EXECUTIVE MODEL
DIET ABSOLUTE RISK
DIET RELATIVE RISK
MORTALITY RISK ADJUSTMENT EXECUTIVE MODEL
TOBACCO BKGD RELATIVE RISK
OBESITY BKGD RELATIVE RISK
ALCOHOL BKGD RELATIVE RISK
PARTIAL BACKGROUND RISK ESTIMATION EXECUTIVE MODEL
LUNG CANCER BKGD ABSOLUTE RISK
OSTEOPOROSIS BKGD ABSOLUTE RISK
CHOLESTEROL BKGD ABSOLUTE RISK
HYPERTENSION BKGD ABSOLUTE RISK
ASPIRIN BKGD ABSOLUTE RISK
DIABETES BKGD ABSOLUTE RISK
DEPRESSION BKGD ABSOLUTE RISK
COLORECTAL CANCER BKGD ABSOLUTE RISK

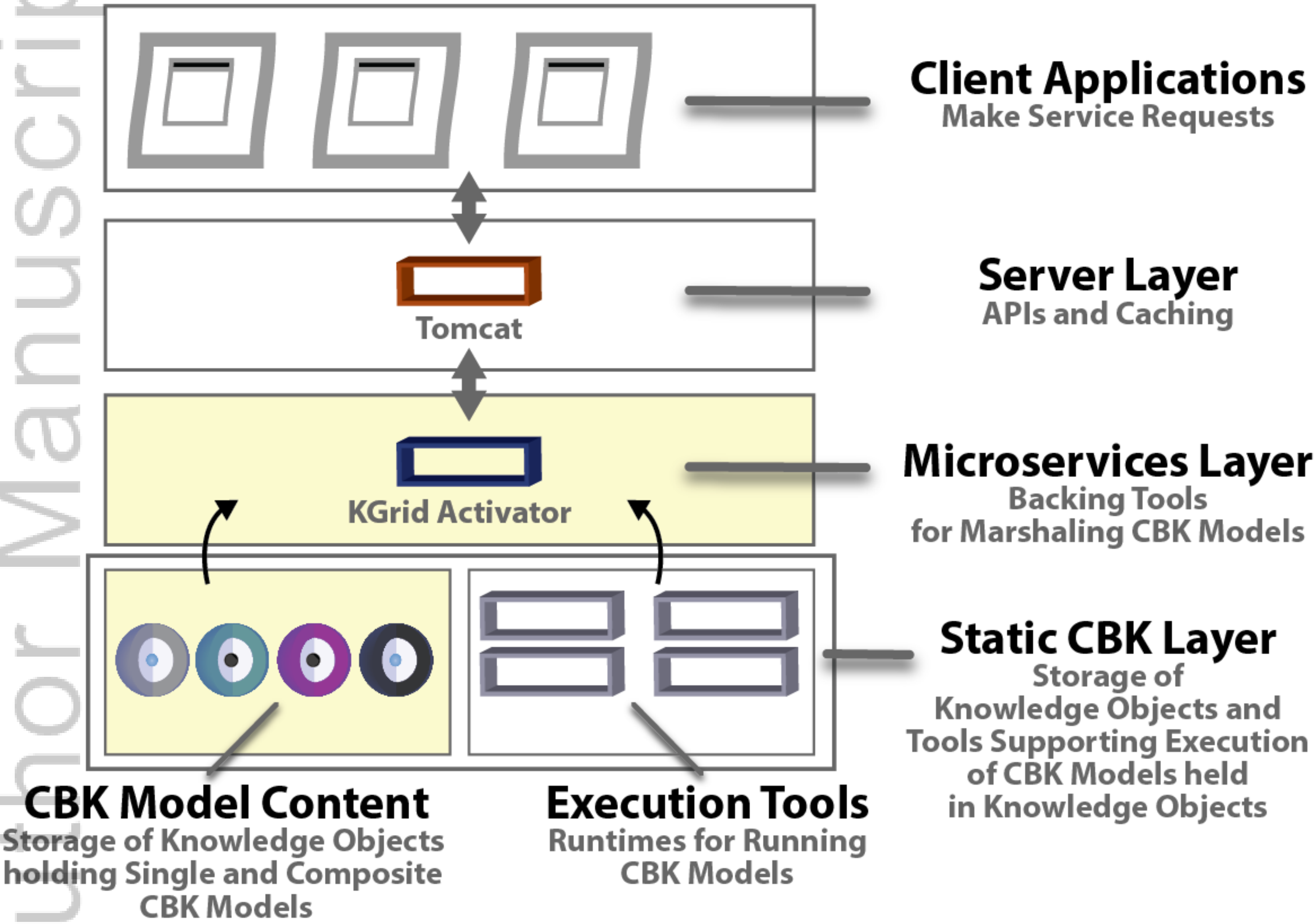
LRH2_10325_ComputeSequence2021.v4.png



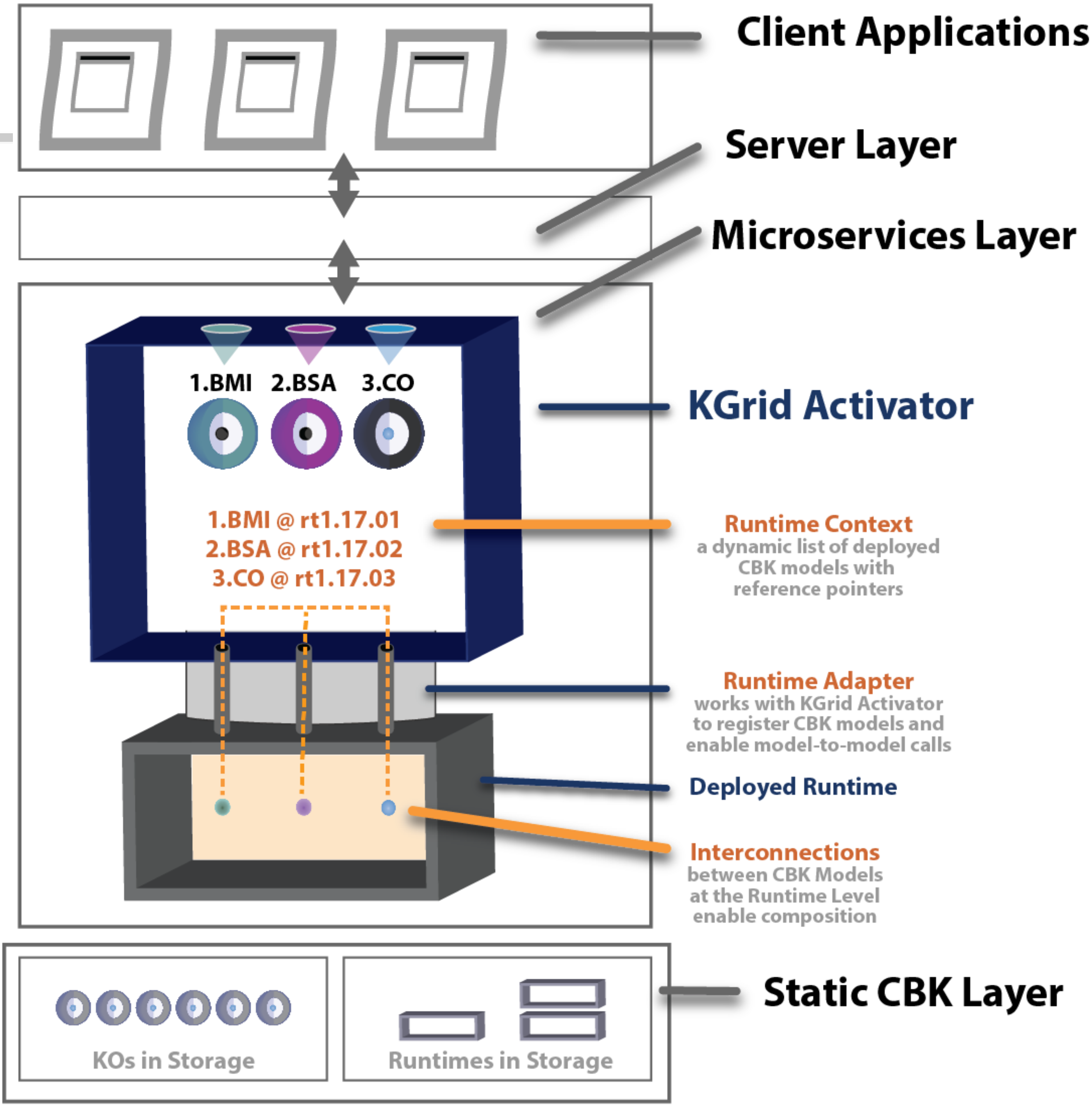
LRH2_10325_GeneralModelComposition2021.v3.png


KO's
CBK Model
Knowledge Payload Bit Sequence
KO's
Deployment Description
Specification of the CBK Model's
Runtime Requirements
KO's
Service Description
Specification for an API for the CBK Model
KO's
Metadata
Descriptions of an Array of
Knowledge Object Properties
KO's
Identifiers
Persistent Identifiers and Versions

LRH2_10325_KnowledgeObjectGraphic2021.v4-01.png



LRH2_10325_OverviewGraphic2021.v2.png



LRH2_10325_RuntimeContextGraphic2021.v4-01.png