The cloud abstraction model delivers a shared pool of configurable computing resources (processors, storage, and so on) that can be dynamically and automatically provisioned and released. Cloud capabilities include three levels of service offerings—software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS)—letting users focus on what the services provide rather than how they’re implemented or hosted.

Current cloud research places substantive emphasis on low-level technological trajectories. For instance, cloud scalability is heavily researched with efforts focusing on service horizontal scaling (that is, adding new server replicas and load balancers to distribute the load) or vertical scaling (on-the-fly changing of the assigned resources to an already running instance) and associated issues such as load balancing. However, we observe far less concentration on other aspects of the cloud, such as cloud application development.

A clear need exists for emphasizing the development of enhanced composite service offerings at the application (or SaaS) level and assigning or reassigning different virtual and physical resources dynamically and elastically. This leads to forming service syndications on demand at any level of the cloud stack that might potentially involve various SaaS/PaaS/IaaS providers by breaking up the current monolithic approach.

Here, we examine such concerns, shedding light on some serious shortcomings in current cloud delivery models and management approaches. We also explain how to achieve holistic cloud solutions on the basis of the cloud blueprinting concept.

The Cloud Delivery Model Landscape
Examining the three cloud delivery models, we observe a recurrent theme. They’re all constrained by the capabilities available via the provider at the delivery level and don’t allow for easy extensibility or customization options. Cloud service developers need better ways to orchestrate a cohesive solution that assembles virtual services when each cloud-stack services tier is possibly provisioned from different cloud providers. To realize this vision, we must tackle several key challenges.

At the IaaS level, it currently isn’t possible to cross-configure the virtual machines comprising a specific cloud service. Moreover, cross-machine configurations must be inferred at deployment time rather than being statically allocated, as is the norm with IaaS solutions today.

Similar problems permeate the PaaS solution space. PaaS offerings are constrained by providers’ capabilities and don’t allow easy extensibility, mashup, or customization options at the consumer or developer levels. The underlying PaaS middleware and the applications built on
it aren't portable across many different public and private clouds and cloud providers. Furthermore, many hosted middleware solutions aren't integrated with any IaaS management and thus lack elasticity and scaling benefits.

SaaS is also predominantly tethered to proprietary application platforms in which the cloud provider runs all elements of the service and presents a complete application to the client. Cloud solution developers can't easily customize SaaS applications at the process level or simply combine them with external services from other providers to offer improved functionality to the client or to other developers.

From the preceding discussion, we can see that current cloud solutions are fraught with problems. First, they introduce a monolithic SaaS/PaaS/IaaS stack architecture in which a one-size-fits-all mentality and vendor lock-in prevails. They don't let developers mix and match services from diverse cloud service tiers and configure them dynamically to address client and application needs. Second, they introduce rigid service orchestration practices tied to a specific resource/infrastructure configuration for cloud services at the application level. These issues hamper the (re)configuration and customization of cloud applications on demand to reflect evolving interorganizational collaborations.

Clearly, developers must be able to mashup services from a variety of cloud providers to create a cloud ecosystem. This type of integration lets developers tailor services to specific business needs using a mixture of SaaS, PaaS, and IaaS. Cloud ecosystems have been termed business process as a service, or BPaaS, emphasizing the focus on enterprise-specific services. BPaaS lets developers create unique end-to-end business processes that are usually syndicated with other external services (possibly provided by diverse SaaS providers). BPaaS requires effective management of cloud resources.

Managing the Cloud

To represent and manage cloud resources, developers usually rely on a metadata or model-driven approach. Metadata constructs, called templates, provide an understanding of the features used to deliver reliable and scalable cloud deployments and achieve better interconnection between virtual and physical infrastructures.

The Distributed Management Task Force (DMTF) uses templates to manage cloud resources of various types and support the cloud service life cycle by, for instance, describing how a cloud offering is presented and consumed. The service offering is abstracted from the specific type of cloud resources offered, and the provider (or developer) uses service templates to describe in a general form what a cloud service can offer. The provider (or developer) then customizes the cloud service template to reflect the fact that the application must create a service offering for one or more specific consumers by including such information as which machines, IP address ranges, and storage requirements are necessary.

Experts have suggested that providers employ model-driven approaches to automate the deployment of complex IaaS services on cloud infrastructure. For instance, researchers have proposed a virtual appliance model that treats virtual images as building blocks for IaaS composite solutions. Virtual appliances are composed into a virtual solution model which then helps developers determine deployment-time requirements in a cloud-independent manner using a parameterized deployment plan. Similarly, other research describes a solution-based provisioning mechanism using composite appliances to automate the deployment of complex application services on a cloud infrastructure.

The aforementioned approaches concentrate mainly on IaaS cloud services. In addition, templates capture static information while being notoriously hard to combine with other compatible templates to render composite offerings.

The blueprint model solution that we describe next, takes into account and furthers such considerations to address cloud solutions that intermix SaaS-, PaaS-, and IaaS-layer service offerings.

The Blueprint Model

Vendor lock-in is viewed as one potential drawback to cloud computing. Analysts recently conducted studies that reveal that vendor lock-in and immature cloud interoperability standards surpass even security concerns as the biggest objection to moving to the cloud. Seen from the perspective of developing new SaaS (or BPaaS)-style applications, this means that if a client (or developer) leases an SaaS application—for example, order management or tax and tariff administration—the client receives a monolithic SaaS stack with little ability to change or customize the application without the original provider intervening. The provider uses its own PaaS/IaaS functionality, or relies on third-party offerings, which are opaque to the client, who has absolutely no control over the PaaS/IaaS offerings that underlie the SaaS application.

The only way to address this serious limitation is to break the cloud monolith and let a client (or developer) who leases an SaaS application freely choose and intermix the lowest-cost and best quality PaaS/IaaS-layer offerings (which different providers might supply) to service it. This approach, which we refer to as cloud blueprinting, converts all cloud-stack layers into general-purpose commodities to attract

Blueprinting the Cloud
partners and build a true cloud ecosystem.

Cloud blueprinting promotes autonomous services (at all levels of the cloud stack) and lets clients or developers combine or swap any service at any layer with a service at the same layer without having to stop and modify components elsewhere in the stack. Such considerations lead to the syndicated, multichannel cloud delivery model illustrated in Figure 1, where we contrast it with the monolithic cloud-stack solutions that permeate the cloud today. The monolithic solutions enforce one-way vertical deployment “channels.”

Blueprinting is a novel, powerful solution that lets BPaaS (or SaaS) applications dynamically run on fully virtualized clouds. It abstracts cloud-stack components to provide a simplified method for provisioning and automating cloud services. It also correlates application-level decisions regarding virtualized end-to-end SaaS services (the BPaaS layer in Figure 1) in a top-down fashion and uses them to drive resource provisioning. These decisions can also help adjust the workload and traffic to automate the dynamic configuration and deployment of application instances onto available cloud resources.

Blueprinting promotes service virtualization — by accommodating a service-oriented-architecture (SOA)-enabled approach — and independent layering for all cloud-stack layers. It also achieves interoperability by exercising a finer level of control over cloud services.

To realize the blueprint model, we must interlace several interrelated components:

- a declarative blueprint definition language (BDL), which provides the necessary abstraction constructs to describe cloud services' operational, performance, and capacity requirements;
- a blueprint constraint language (BCL), which specifies explicitly stated rules that prescribe any aspect of cloud service including policies, performance, and capacity requirements; and
- a blueprint manipulation language (BML), which provides a set of operators for manipulating, comparing, and achieving mappings between blueprints.

Figure 1. Conventional vs. syndicated multichannel cloud delivery model. The conventional, monolithic cloud solutions enforce one-way vertical deployment “channels.”
defined in BDL. This item also includes a simple blueprint query language for querying blueprint collections.

Let’s briefly examine these three elements of the blueprint model.

**The Blueprint Definition Language**
To understand blueprinting, consider an external developer (for instance, a virtual service operator or provider) that offers mashup solutions bundling together different types of interactive SaaS telecommunications services (wireless home broadband and multichannel video programming services, including video-on-demand) and enterprise-grade backend services (rating and broadband billing for cable TV or fixed broadband services, billing processes, invoicing, accounts receivable, and so on). The developer implements the turnkey service solutions (BPaaS), which might involve services from multiple providers, in an end-to-end fashion.

Service providers publish their offerings — in this case, video-on-demand or interactive TV/open cable applications — using a blueprint model specified in BDL for a telecommunications marketplace. Developers, such as a virtual service provider, can then choose offerings and compose, extend, and customize this blueprint model to develop full-featured service-based applications.

A blueprint model defined in BDL is based on a clear separation of service processing concerns and is minimally distilled in the following interrelated templates:

- **Operational service description** contains the functional characteristics of a service, such as service types, messages, interfaces and operations.

- **Performance-oriented service capabilities** include quantifiable key performance indicators (KPIs) associated with services. Typical examples are response-time ranges, throughput, delivery, latency, bandwidth, mean time between failure, mean time to restore service, and so on.

- **Resource utilization** describes the physical infrastructure and resources required to run a specific service. In general, this template expresses the workload profile, including average and peak workload requirements. For instance, a service provider might declare specific technical features that it must take into account for its service to operate properly, such as the server (disk I/O and network) bandwidth required for true on-demand delivery of streaming media. This template can express information packaged using the DMFT’s Open Virtualization Format (OVF).4

- **The policies template** prescribes limits, or specifies any aspect of a business agreement necessary to use a particular service, and includes security, privacy, and compliance requirements.

Information contained in these templates must be cross-correlated to deliver sound cloud application solutions on the basis of the blueprinting approach.

**The Blueprint Manipulation Language**
The blueprint model exposes its information in a manner that facilitates comparison and the simple composition of blueprints to express end-to-end offerings from various providers. BML is based on model-management algebraic operators, such as match, merge, compose, extract, delete, and so on. These operators accept source blueprint templates as input and return a new blueprint template as result.

To exemplify BML’s use, consider using the merging operator on four blueprints describing high-definition IPTV, video on demand, broadband Internet, and voice-over-IP services to create an end-to-end triple-play service. This operator will yield a target blueprint model that aggregates all four blueprint models by defining mappings between them. The operator will ascertain that it can match the four models to each other and that the target blueprint model doesn’t violate constraints or capacity requirements.
Supporting the Cloud-Service Life Cycle

Cloud services are available via a cloud marketplace, an efficient online platform that manages service distribution and bid management, in which providers store their offerings and clients can discover and deploy third-party cloud applications that they can integrate with their own. The concept is similar to that of the Google Apps Marketplace. To understand how the cloud blueprint model facilitates this process, consider Figure 2.

After a provider (or developer) has created all the components of a discrete service, it uses BDL to describe all its relevant aspects in a structure called a source blueprint, which it stores in the marketplace. The provider or developer also customizes the source blueprint templates to create a service offering for different types of consumers. Each such offering is accompanied by an SLA to delimit the range of service instances defined in the offering. This happens during the blueprint model's design phase.

Now consider a virtual service operator who wishes to bundle together several interactive telecommunications services to produce a turnkey application. This developer must first discover these discrete services' source blueprint models. This happens during the selection phase Figure 2 shows. Subsequently, the developer creates an interim target blueprint model by combining the set of source blueprint models it selected. For this purpose, it uses the BML operators. At this point, the interim blueprint model generates a deployment plan, which might drive PaaS resources and determine virtual machine placement and network configuration. The developer could, for instance, choose to deploy different PaaS options to address customization...
requirements. The application might deliver the same content over diverse devices, such as TVs, PCs, and smart phones, and provide discounts to certain customers during low network-utilization periods. PaaS services for this type of application could involve workflow facilities, event-processing functionality, deployment, and hosting. As usual, not all these services must come from the same third-party PaaS provider.

The deployment plan matches the service-based application’s demands, addresses scalability estimates, and, in general, tries to optimize the service assembly’s performance according to quality-of-service requirements in the interim target blueprint model. This helps to provision resources and adjust the workload and traffic during the deployment phase. It also provides upstream and downstream alternatives to automate the dynamic configuration and deployment of application instances onto available cloud resources.

Finally, during the testing and monitoring phase, the developer gradually refines the abstract information contained in the interim target blueprint to reach the level of rigor and concreteness required for a production-ready cloud application. This results in an optimized target blueprint model describing the integrated application. The developer then publishes this blueprint model in the marketplace repository for potential clients to discover and use.

One of the greatest challenges facing longer-term adoption of cloud computing is the ability to automatically provision services, effectively manage workload segmentation and portability (that is, the seamless movement of workloads across many platforms and clouds), and manage virtual service instances, all while optimizing the use of cloud resources and accelerating the deployment of new services. Within such a cloud environment, it’s also important to equip developers with a unified approach that lets them develop cloud applications on top of existing applications at any layer of the cloud stack from multiple cloud providers. The cloud blueprinting approach we present here greatly helps cloud application developers deal with such challenges.

Acknowledgments
We’re grateful to Luis M. Vaquero of HP Labs for his valuable comments and suggestions. The research leading to these results has received partial funding from the European Community’s 7th Framework Program under the Integrated Project 4CaaST, grant agreement number 258862.

References

Michael P. Papazoglou is the chair of the Computer Science Department at Tilburg University, and the scientific director of the European Research Institute in Service Science and the EC’s Network of Excellence in Services, S-Cube. His research interests include service-oriented computing, cloud computing, distributed computing, business process management, and large-scale data sharing. Papazoglou has a PhD in microcomputer systems engineering from the University of Dundee, Scotland. He’s a Golden Core Member and a Distinguished Visitor of the IEEE Computer Society. Contact him at mikep@uvt.nl.

Willem-Jan van den Heuvel is a full professor in computer science at the Department of Information Systems, Tilburg University, managing director of the European Research Institute in Service Science, and program director of the Erasmus Mundus Joint-Degree Master Program on Service Science. His main research interests revolve around (cloud) service engineering, performance analytics of software-enabled service networks, and business transaction management. Van den Heuvel is author of Aligning Modern Business Applications and Legacy Systems: A Component-Based Perspective (MIT Press, 2007). Contact him at wjheuvel@uvt.nl.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.