# Web Systems - Assignment

James Donohue - james.donohue@bbc.co.uk

## Contents



# 1 Business and technological context

*mymuesli* is a German company founded in 2007 that sells custom-mixed muesli by post in four European countries. This paper uses *mymuesli* as a case study of how the Internet and information systems (IS) have the capacity to transform every part of the enterprise, and indeed to be integral to how it achieves its objectives.

## 1.1 e-Business models

Papazoglou and Ribbers (2006) define an e-Business model as "descriptive representation of the fundamental components of a business that operates partially or completely on the Internet". A detailed discussion of features common to all business models, not just ones for e-Business (such as value proposition and revenue model), is beyond the scope of this paper. However, using the authors' classification system for such models, we can see that *mymuesli* is an 'Internet-enabled' business and falls specifically within the business-to-consumer (B2C) category of 'e-Shop' proposed by (Timmers 1998).

The company produces a 'niche' product which may have been prohibitively expensive to sell and distribute prior to the Internet. By operating as an e-Shop, *mymuesli* gains a "low-cost route to global presence" (Timmers 1998), meaning that company has the potential to be profitable selling its product direct to consumers around the world without the need for costly intermediaries. Another way of putting this is to say that the Internet allows manufacturers to "compress the distribution channel" (2010).

On the other hand, although the Internet makes the physical distance between *mymuesli* and its customers less of a problem, it creates other kinds of challenges, such as increased competition and the relative ease with which a customer can abandon their 'shopping basket' and buy from a competitor instead (low customer switching costs, in Porter's terms (2008)). There is also an increased need to focus on online marketing and search engine optimisation (SEO) to ensure that customers find the product in the first place.

It is notable that since 2012 *mymuesli* has also operated two physical stores in Germany (mymuesli GmbH 2017), putting it in an unusual category, along with others such as Amazon (McCarthy 2017), of a company that initially only operated online but later opened high street shops and raises the question (originally framed around traditional retailers embracing the web) of finding the right "mix of bricks and clicks" (Gulati and Garino 2000). One goal of this stategy is be to build customer awareness of the brand and provide more promotion opportunities.

Furthermore, should *mymuesli* enter the hotel market its business model will also encompass business-to-business (B2B) interactions. The idea of allowing a hotel chain to offer a custom breakfast "by mymuesli" is an instance of 'co-branding', which not only exposes *mymuesli*'s offering to a wide range of potential customers but also "enhances the credibility of the hotel's brand by borrowing credibility from other brands" (Yip 2005). Combined with an Application Programming Interface (API) for receiving and fulfilling hotel orders, this would represent a step towards a Value Web e-Business model based on inter-company-relationships that exploit the possibilities of new technology (Papazoglou and Ribbers 2006).

## 1.2 Role of information systems

Information systems (IS) now have a major role in most businesses, but they are even more critical to businesses that conduct the majority of their activity online such as *mymuesli*. For Stair and Reynolds (2014) "information is one of an organisation's most valuable resources", and a key function of an organisation's information systems is to collect and manipulate data and transform them into information which can provide value to the organisations' decision-makers.

In the case of *mymuesli* data might include raw facts such as a customer postal address, stock levels for a particular muesli ingredient or an individual order number. By creating relationships between these data valuable information can be generated, such as the number of orders ready for shipping to a particular postcode, or an estimated date when stock will run out (based on historical orders). As well

as allowing management to monitor the business effectively, this information can also be used as a basis for prediction. The use of computer-based information systems (CBIS) would therefore allow *mymuesli* to anticipate demand by ordering new stock automatically and leverage economies of scale by batching up orders for delivery to the same area. This 'feedback mechanism' within such systems is what allows the business to adapt to changing market conditions and "is critical to helping organisations achieve their goals, such as increasing profits" (Stair and Reynolds 2014).

## 2 Solution architecture

Figure 1 gives a high-level breakdown of regular business functions needed by *mymuesli*. Note that some functions that might only be performed occasionally are omitted (for example, legal advice). Also note that functions like IS 'cut across' other areas, providing services to the whole business, whereas others are more self-contained (e.g. Purchasing). Nor does this figure directly imply a human resource (HR) structure - in a 'start-up' context multiple functions might be performed by the same person.

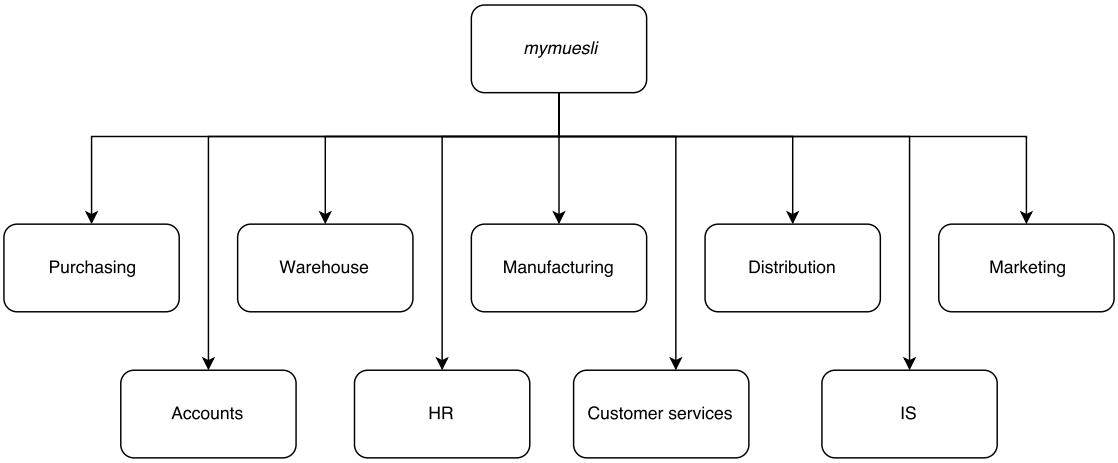


Figure 1: Breakdown of *mymuesli* business functions

We can further analyse the day-to-day operation of the business by identifying some of the most important business processes and modelling them using BPMN. Figure 2 outlines the process involved in preparing a customer order for shipping once received via the website. This process is a good candidate for initial study because it is fundamental to *mymuesli*'s value chain - without it there is no product to sell. Interactions with third-party web services are identified as 'service' tasks using the gear icon.

The process given here assumes a just-in-time (JIT) approach to managing inventory where ingredient stock is ordered only as it is needed, reducing the costs and waste associated with holiding excess inventory. This is based on the assumption that *mymuesli* has reliable suppliers and steady production (The Economist 2009). One limitation of Figure 2 is therefore that it does not show how to handle a situation where stock delivery is delayed or there is a production backlog, which would lead to the shipment needing to be cancelled.

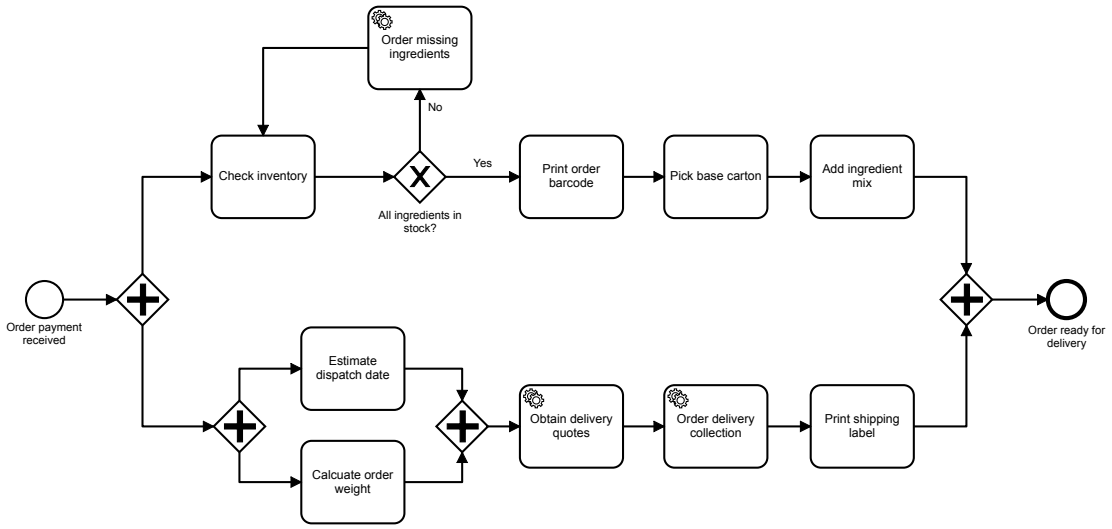


Figure 2: Process for order fulfilment described using BPMN

## 2.1 Microservices

In order to anticipate the need for scaling and give *mymuesli* the greatest flexibility as it grows, we suggest an architecture based around discrete microservices which can be developed and deployed independently (Fowler and Lewis 2014). At the price of increasing the overall complexity of the solution, this enables various combinations of current and future requirements (B2C, B2B, etc.) to be fulfilled by composing together different services.

Once a system boundary is determined, business process descriptions like the ones above and user stories can be used to help identify the services that may be needed. Although this can be done by business area, another common approach is by entity, by looking for the nouns used in user stories, for example:

- Orders
- Inventory
- Customers

Figure 3 shows how such services could then be used by a range of front-end interfaces to satisfy different user stories The boxes in light grey represent APIs provided by third parties, such as a postal delivery broker service.

As with most tiered approaches to architecture, the main benefit of implementing business logic in APIs (rather than embedding it in the website code) is that it facilitates reuse: the figure shows how a native mobile app could easily be developed using the same APIs as the website.

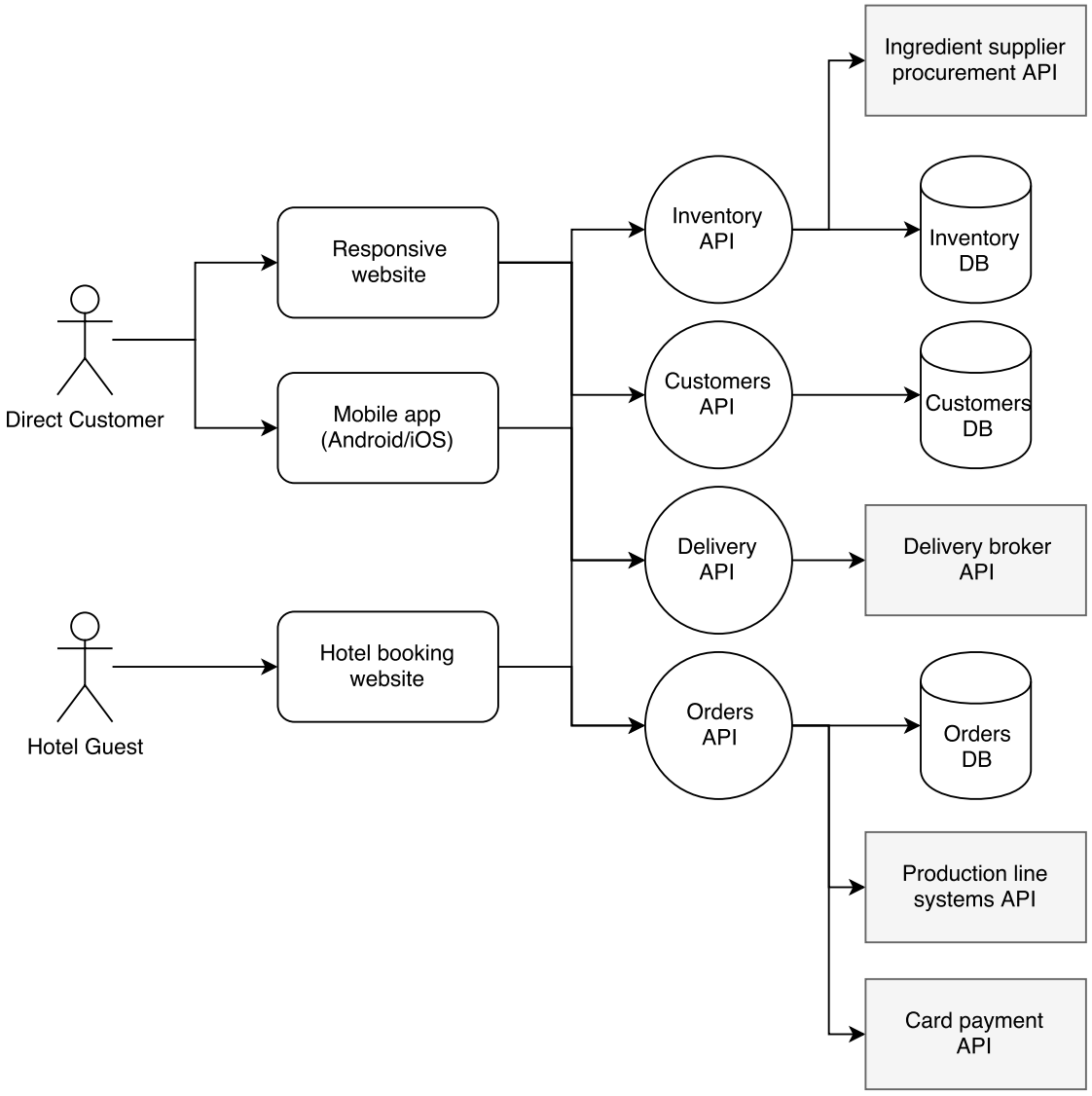


Figure 3: Microservices architecture for *mymuesli* showing different front-ends

# 3 Discussion

## 3.1 Business processses

### 3.1.1 Modelling processes using BPMN

Business process modelling is a form of analysis that describes the dynamic activities performed by a business. One objective of creating a process model is to gain insights into how a business operates, which once validated can be used during the design phase of an information system to assist in the creation of further artefacts such as a data model (van der Aalst and Stahl 2011). For example, when a customer is ready to place an order with *mymuesli*, the company must acquire the user's postal address in order to know where to dispatch the order to when ready, either by retrieving it from information system storage if the user has previously logged in or asking the user to provide it at checkout. This entails that any data/object model for the customer must include all required address information.

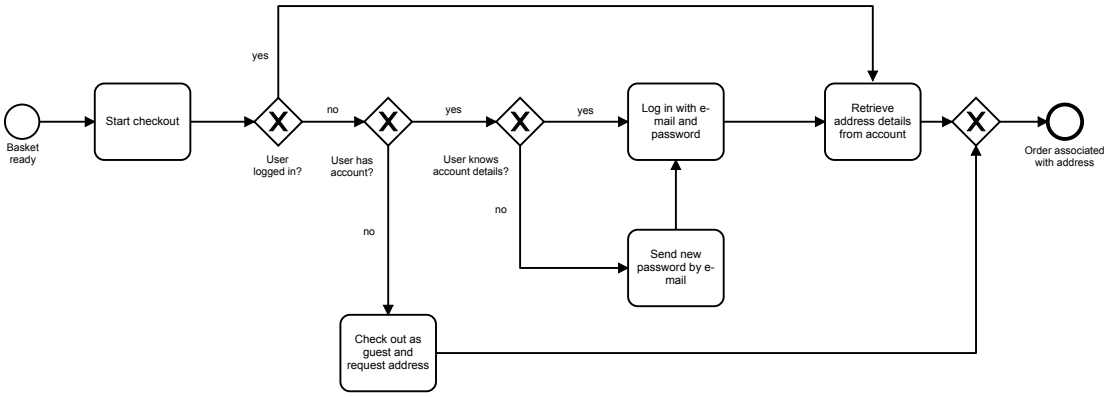


Figure 4: Business process for associating an address with an order in BPMN

There are a number of normative visual languages available for describing businesses processes. One example of these is Business Process Modelling Notation (BPMN), which uses defined symbols for events, tasks, flows and gateways to model business processes. Figure 4 shows such a BPMN diagram representing the process for a postal address being associated with an order. Although BPMN has some similarities with the UML activity diagrams familiar to many working in software development (OMG 2017), one aim of BPMN is to be intelligible by a wide range of business users and not just developers (Papazoglou and Ribbers 2006). On the other hand, one weakness of BPMN in contrast to earlier approaches such as 'classical' Petri nets is its lack of formal semantics in some areas, meaning that BPMN diagrams "typically lack the precise mathematical basis that is required to render them really unambiguous" (Kossak et al. 2014). Even with this caveat, it remains a useful high-level tool for intra- and inter-business communication.

### 3.1.2 System boundaries and outsourcing

Papazoglou and Rivers (2006) suggest that, by making 'business rules' visible, business process models make it easier for companies to adapt in changing market conditions. This advantage is likely to be of particular relevance to retailers such as *mymuesli* that operate primarily in the rapidly-changing online environment.

In order to employ a systems approach to modelling business, a necessary first step is to identify the system boundary. An analogous step is required in the Unified Process, where the requirements analyst must determine what is part of the system (i.e. inside the system boundary) and what is in its environment before use case modelling can begin (Arlow and Neustadt 2005). For the current study, the system boundary encompasses the *mymuesli* online shop, the manufacturing process ...

*mymuesli*'s competitive position can be evaluated within the framework of Porter's five forces (2008) in order to formulate a strategy. As it sells a niche, bespoke product the threat of substitutes is relatively low, partly due to the high costs of entry for competitors in replicating the required customisable production line and online 'mixer' interface. This means there are currently no direct competitors. However, suppliers of raw ingredients are relatively powerful and can pass on rises in the price of crops (for example) to *mymuesli*, and customers may become price sensitive toward the product.

This suggests that *mymuesli* should guard against increasing supplier power further by not being dependent on external suppliers of IT services that can increase their prices and cannot be easily replaced.

From a resource-based view of e-Business, a company's resources and competences are what lead to value creation and differentiate the company from its competitors. This implies that innovative companies should consider which functions are core to their business and which can be left to others, particularly where they are performed on too small a scale to be economical. For Papazoglou and Ribbers, "strategic outsourcing of all activities that are not core competencies potentially allows a company to focus on all those areas which add most value" (2006). This echoes Peter Drucker's famous maxim of "do what you do best and outsource the rest" (Forbes 2010).

Obvious core competencies for *mymuesli* are the sourcing and selection of appealing muesli ingredients, mixing and packaging the muesli, and deciding how and where to market the website. Non-core activities that are candidates for outsourcing include delivery of orders to customer addresses, accounting and human resources (HR), copywriting, graphic design, processing credit card transactions and supporting the servers that run the website. However, the choice is less clear-cut in some areas: is writing the website application code a core competency, or something that could be handled by a managed e-commerce platform such as Shopify?.

Some of this is summarised in Table 1.

Table 1: Main functional areas and activities for *mymuesli* .

| Functional area | Example activities and capabilities |
|---|---|
| Purchasing | Managing relationships with suppliers of ingredients, packaging, etc. |
| | Negotiating best possible prices |
| | Placing orders at the correct time to avoid being 'out of stock' |
| Warehouse | Ensuring ingredients are properly stored |
| | Maintaining ingredient stock inventory |
| Accounting | Corporate tax return, VAT, payroll |
| Human resources | Performance management, case work, salary and grading |
| IT | E-mail, anti-virus, web access |

#### 3.1.2.1 RPC from SOAP to REST

Remote Procedure Call (RPC) is "an interprocess communication (IPC) mechanism that enables data exchange and invocation of functionality residing in a different process" (Microsoft 2003). Such in-

vocations are normally synchronous request-response interactions (meaning that the calling process 'blocks' until a response is received), and presented as if they were normal local procedure calls, thereby abstracting the details of communicating with the remote process from the developer; popular implementations include CORBA, Java's RMI and .NET remoting (Hohpe and Woolf 2004). The first wave of web services adopted a similar approach, implementing RPC-style interactions on top of SOAP (and normally HTTP, although SMTP and JMS bindings for SOAP are possible). Unlike the implementations listed above, SOAP is an open standard and uses XML to encapsuate a method call and its response in a platform-independent way, with the goal of being both simple and extensible (W3C 2007). It is complemented by technologies such as Web Service Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI), which allows SOAP clients to discover what services are available, where to find them and what operations they support - providing an explicit 'contract' of behaviour for clients to code against.

By contrast, REST grew in popularity in the early 2000s as a more lightweight and less verbose alternative to SOAP that leans more on the inherent statelessness and semantics of HTTP methods (e.g. `PUT`, `DELETE`) and a uniform interface based around URIs to provide web service capabilities. Another design goal of REST was to leverage the caching capabilities of HTTP to provide improved scalability for web services (Fielding 2007).

Despite this, most RESTful web services are still implemented through the synchronous request-response paradigm of RPC. Moreover, as an 'architectural style' instead of a protocol, REST lacks features provided by SOAP such as built-in mappings of data types to byte sequences or standardised error handling. SOAP also benefits from considerable support in enterprise tooling and automation that is only now becoming available in REST.

For the *mymuesli* case we can assume that point-to-point HTTP communication between endpoints is practical so SOAP's support for alternate transports is not vital, and as the different services will be developed by the same team the benefit of rigid WSDL-style specification is reduced.

#### 3.1.2.2 Service-oriented and microservices architectures

Service-Oriented Architecture (SOA) is an approach to designing systems where specific functions (such as managing customer details) purposely implemented as networked services that can be used by other applications, allowing a complex (and perhaps distributed) information system to be built in a more modular fashion out of interconnected parts that can be developed and modified independently (Stair and Reynolds 2014).

Depending on the area of functionality these services cover, a SOA may share some characteristics with a microservices architecture. In the approach made famous in the web sphere by companies like Netflix, microservices are usually RESTful rather than SOAP-based and highly granular, aiming to "do one thing very well" (Stair and Reynolds 2014). Microservices generally aim for an organic 'bottom-up' approach rather than a 'top-down' model where all service interfaces are architected by a central team and and commonly use middleware such as enterprise service bus (ESB) as an integration point.

One challenge with a microservices approach is creating a machine-readable description of services to allow for discovery and negotation by clients. Some authors consider the need for a service directory and a description of each service's interface to be central to SOA (Hohpe and Woolf 2004). While SOAP and WSDL support these requirements in the enterprise SOA context, REST has no such built-in capabilities, and although equivalents for RESTful services (such as as WADL and Swagger, now known as the OpenAPI Specification) have emerged (SmartBear 2017) they still do not have the same level of support.

#### 3.1.2.3 Heterogenous infrastructure

According to Hohpe and Woolf (2004), "creating a single, big application to run a complete business is next to impossible", despite the efforts of some Enterprise Resource Planning (ERP) vendors such as SAP. This means that it is unrealistic to expect an existing commercial off-the-shelf (COTS) system to implement all of a business' needs, or to write such a system from scratch. Using multiple different applications has the benefit of allowing businesses to use a 'best of breed' approach in selecting the right software for each function. However, where companies have such pre-existing IT infrastructure the need for integration between systems arises. Enterprise Application Integration (EAI) attempts to solve this problem through middleware providing features such as messaging and workflow management (Hohpe and Woolf 2004).

For the purpose of this study we can imagine *mymuesli* as a 'green field' project without any 'legacy' IT infrastructure. However it is possible that the company would still choose to buy some existing off-the-shelf software to provide certain functions, such as accounting, and may aquire diverse information systems as a result of acquisitions or mergers (Stair and Reynolds 2014). In this scenario the company might consider implementing an EAI suite to facilitate the integration of these systems.

#### 3.1.2.4 Beyond RPC: Loose coupling and message-oriented middleware

One important aspect of designing information systems is the need to accept that change is inevitable, whether to business requirements or within the systems that provide different business functions, even as part of normal maintenance. For example, upgrading a stock control system to the latest version could break interoperability with a linked accounting system. Moreover, faults occur and distributed systems may become unavailable for periods of time. Although some Remote Procedure Call (RPC) implementations allow synchronous remote requests to be made by simply calling a method, for Hohpe and Woolf (2004) the idea of abstracting RPC using the semantics of a local method call is 'asking for trouble' due to the increased unpredictability involved.

Loose coupling is the name given to the design goal of reducing number of assumptions that systems make about each other in order to increase tolerance to change and failure. Loose coupling (also a design goal of SOA) advocates using platform-independent, self-describing data formats and (in an EAI context) eliminating the temporal dependency by rethinking remote calls into self-contained asynchronous messages that are sent to a logical 'channel', an approach usually implemented through message-oriented middleware, or MOM. Such systems may also offer the ability to translate between message formats and route messages to different places depending on their contents, along with a system management and monitoring functions (Hohpe and Woolf 2004). Despite increased flexiblity, the downside of these approaches is that they can be more complex to design, implement and debug.

## 3.2 Impact of mass customisation on production

Although it also offers pre-configured 'off-the-shelf' muesli mixes, the ability to offer a customised muesli is integral to *mymuesli*'s value proposition, as indicated by the company's name. This reflects the trend towards mass customisation, the intent to "provide products and services that best serve customer needs while maintaining near-mass production efficiency" (Tseng and Jiao 2001). Such an approach means that manufacturing and distribution processes must be designed with customisation in mind rather than as an afterthought. This is because, as Pine (1993) points out, one key requirement for business processes designed to support mass customisation is that they are *instantaneous*, meaning they can be linked together as quickly as possible. Therefore a design goal of the manufacturing process

must be for it to rapidly assemble any combination of muesli ingredients for a given order in the correct quantities with the minimum of errors.

One helpful step is is to reduce the number of possible combinations of ingredients. Although the *mymuesli* website 'mixer' shows 76 options, in fact all mixes must include one of the 14 standard muesli bases such as 'Five Grains'; this corresponds to the 'common base' within the product family architecture described by Tseng and Jiao (2001). These bases can be pre-filled into cartons (according to their projected popularity) prior to any actual order being received, leaving only 62 actual custom options and reducing the number of potential permutations significantly. Further time savings can be made by pre-mixing combinations of ingredients that are frequently ordered together.

In order to minimise human error (and therefore waste) as far as possible, a fully automated mixing apparatus is envisaged that routes different ingredients from bulk containers and adds them to the carton for each order. The flow of material into the mixer is stopped when an electronic scale underneath the carton senses that the correct quantity has been added, with stock levels also being updated automatically. A number of cartons could be filled in quick succession by being placed on an automated guided vehicle (AGV) that moves under a series of suitably arranged storage bins; note this is the approach used by the ReciPure system (Zeppelin GmbH 2016).

### 3.2.1 Tracking systems

Another important tool for automating the production process is barcodes, an example of machine-readable data input (Stair and Reynolds 2014). If *mymuesli* attaches a barcode containing a representation of the order number to each carton at the start of production, and scanned it repeatedly, the location of the carton and therefore the order's progress can be tracked with relative certainity.

On the production line, by reading the barcode and looking up the order details, the mixing apparatus can know how much of each ingredient to dispense. Similarly, when orders are collected for distribution by courier, the barcode can be scanned to trigger the status of the order to automatically be updated to 'shipped'. In this way barcodes (including 2D variants such as QR Codes) assist with traceability of orders and reduce the scope for errors. Recording a timestamp along with each scan allows for analysis of how much time orders are taking prepare. A smartphone app offers a relatively simple and portable way of integrating barcode functionality into production processes: by using the phone's camera, staff can access or edit data about the order wherever they are.

## 3.3 Web front-end

It is apparent that like most e-shops this company requires a web front-end which allows customers to order its products effectively. However, unlike online merchants such as Amazon, which sell a wide range of standardised products with limited options (such as different sizes for a t-shirt) through a mostly standardised interface, the composition of a *mymuesli* order must be tailored extensively by the customer.

This therefore requires either the creation of an bespoke web interface, or the use of an e-commerce platform designed with customised products in mind.

The following are some typical features that any web shop front-end would be expected to provide:

**Account management** Allows customers to create an account, update their address, change their password, etc.

**Search** Enables the discovery of products by allowing customers to search for a particular ingredient or mix by name

**Shopping basket** Lets the customer temporarily store a mix before placing an order (e.g. while they add other mixes)

**Order tracking and management** Allows the customer to check the status of their order and review previous orders

**E-mail notifications** Sends e-mails to the customer once their order is received, shipped, etc.

**Support interface** Allows users to send other questions associated with their account/order, as well as general feedback, and receive a timely response

Since the above features, with some variations, are common to most online shops, *mymuesli* could save duplicated development effort and therefore reduce costs by using existing implementations, either by licensing a commercial e-commerce system or an e-commerce Software as a Service (Saas) platform.

### 3.3.1 Multi-platform strategy

In order to reach the widest possible market, *mymuesli* should deliver e-commerce interfaces that work well for mobile web users as well as desktop browsers. This is because mobile ('m-commerce') is expected to account for 45% of online sales by 2020, with users spending most of their time in mobile browsers instead of apps (Business Insider 2017). This means that any e-commerce business needs to have a strategy for ensuring their site is compatible with the widest range of web-enabled devices.

One thing that makes the web an unusual platform for application development is the relative lack of control that developers have over the user experience. At the application layer client-server interactions over HTTP are well defined (IETF 2014), and the basic hypertext model of HTML is widely understood, but the wide variance in client feature support and web browser vendor-specific 'quirks' have made it historically a challenging environment. With the increase in mobile browser usage, web designers are creating 'mobile-first' designs that must also work well on desktop, while the engineering focus has moved away from creaing specific mobile versions to a single 'responsive' website that embraces the variation in web-enabled devices, typically using media queries (W3C 2012) to allow suitable stylesheets to be presented to different users. Furthermore, since for mobile users temporary loss of network connection is a regular event, *mymuesli* needs to ensure that its m-commerce interface recovers gracefully from such errors in order to avoid losing sales.

#### 3.3.1.1 COTS e-commerce systems

Commercial-off-the-shelf (COTS) e-commerce platforms provide businesses with a range of ready-made e-commerce functionality that may combine a web storefront with an internal order management interface, often with a high level of flexiblility. For example, IBM WebSphere Commerce allows developers with the requisite knowledge of Java, JSP and XML to create custom user interface 'widgets' and back-end integrations (IBM 2010). These platforms may also cater to 'omni-channel' models (including in-store and call centres as well as e-commerce), and can even communicate with legacy mainframe fulfilment or stock control systems via an Enterprise Service Bus (ESB). They are often (though not necessarily) installed on on-premises hardware to allow the necessary integrations to be implemented.

On the other hand, some e-commerce platforms allow the creation of 'plugins', modules of reusable code which add functionality to the platform or customise its behaviour, by means of a defined plugin interface (API) provided by the platform. This enables third-party developers to write and sell plugins that address recurring requirements not provided by the base platform, more mature platforms even

provide an app store to make buying and installing these plugins easier and also providing another revenue stream for the vendor.

The mixer interface needs to implement a number of business rules which are specific to *mymuesli*, for example:

- all mixes must include exactly one of the standard base ingredients
- there is an upper limit on the number of ingredients that can be added (due to finite carton size)
- each muesli should be able to have a custom name and description which will be printed on the tube

In addition, the online shop includes very specific features which provide further feedback to the user as they customise their museli, improving the overall user experience, such as a chart of nutritional information, a list of raw ingredients and other visual hints indicating the flavour profile and other characteristics of the mix (e.g. 'high fibre'). All of this points towards the need for the development of a bespoke interface.

## 3.4 A business-to-business (B2B) perspective: hotel services

One way that business strategists can maximise profits and ensure sustainability is by increasing product accessibility (Porter 2008). By providing breakfast services to hotels, *mymuesli* could open a new channel to customers and raise awareness of its products, however this comes at a cost of the additional complexity of implementing the B2B interface, negotiating service level agreements (SLAs) etc. and also increased risks arising from the introduction of a third party (hotels) to muesli sales.

## 3.5 Other cross-functional concerns

### 3.5.1 Quality assurance

One key aspect of trading is ensuring the quality of the product and/or service being received by the customer, and e-commerce is no exception. Suitably-trained staff working in the *mymuesli* production facility should be able to randomly sample mixes prior to shipping to ensure their contents and the carton label match the order details received via the website, with the information quickly displayed on a smartphone by scanning the barcode. Any discrepancies or failure to meet the required standard can be logged by the system so that managment have visiblity of failure rates and can respond quickly in the event of a spike in problems.

The same approach could be used to capture the number of complaints received by customers via the support interface and categorise them (e.g. delivery problems, quality issues, website errors) in order to priorities areas of the business process in need of improvement (the feedback mechanism described earlier).

### 3.5.2 Security implications

It is essential for the integrity any e-Business that security is built into any technology solution from an early stage (following 'security by design' development principles). Although the need for information security measures around hardware and other infrastructure (in terms of tools such as firewalls and intrustion detection systems) is well known, the need for good application-level security is also critical. *mymuesli*'s website (and any publicly-accessible web services) should be built on the principle of

least privilege, meaning that each component or agent within a system is granted the lowest level of permission necessary to carry out its function.

In the context of the web, any e-commerce site is potentially a target for would-be attackers wanting to intercept credit card information or access confidential customer data. Standards such as the Payment Card Industry Data Security Standard (PCI 2017) exist to ensure that sites using credit card data take necessary security measures and thereby help and create a safe environment (based on enforcing correct use of public-key cryptography standards such as HTTPS/TLS) for e-commerce to take place. As well as confidentiality, the question of availability is also relevant to security. In order to be able to continue to receive orders, *mymuesli* needs to ensure it has sufficient network defenses to remain available in the event of a denial-of-service (DOS) attack.

# 4 Conclusion

*mymuesli* is a worthwhile example to study in the context of web information systems, partly because its business model is founded upon on web technology but also because it demonstrates how startups without significant investment in legacy enterprise systems have the freedom to adopt an innovative information systems strategy. The internet has enabled the development of new business models (such as SaaS) and accelerated the move towards outsourcing of non-core functions, while the explosion of pay-as-you-go cloud-based hosting has democratised access to infrastructure that was previously only available to large corporations.

The emergence of microservice architectures is particularly significant in this context. Although it is less of a revolutionary concept than an evolution of existing patterns - some see its roots in the Unix philosophy (Fowler and Lewis 2014) - along with the RESTful style it continues to challenge a lot of the traditional assumptions about how business information systems are developed and integrated. This can be seen clearly from its adoption within the BBC's own systems.

## 4.1 Relevance of study to the British Broadcasting Corporation (BBC)

On the surface *mymuesli* and the BBC are very different entities. Instead of a physical product, the BBC's products are in the form of services (e.g. Radio 2, iPlayer, the BBC Sport website) and instead of directly charging customers for the services they use a statutory per-household licence fee is charged for owning a television. Nonetheless the BBC could adopt several aspects of this study's approach, for example:

- using BPMN to describe the steps in the business process that occurs when a journalist publishes a news story on the BBC website, in order to make the process visible to management
- with analysis of the streaming TV industry (including recent entrants such as Netflix and Amazon) to position itself where Porter's 'five forces' are weakest to ensure its long-term survival
- break down its legacy monolithic information systems into small self-contained microservices with their own release schedule in order to increase rate of development and tolerance to change
- provide an API to allow third parties such as other broadcasters to submit requests for content from the BBC archive to be transcoded to required formats and delivered by file upload or on physical media

Moreover, the BBC could adopt a strategic approach to its information systems, outsourcing functions as needed in order to focus on its core competencies.

# References


Arlow, J. and Neustadt, I., 2005. *UML 2 and the Unified Process. Second Edition*. 2nd Edition. Upper Saddle River, NJ: Addison-Wesley.

Business Insider, 2017. *Mobile sales drive unexpected UK e-commerce growth* [online]. Available from: http://uk.businessinsider.com/mobile-sales-drive-unexpected-uk-e-commerce-growth-2017-1 [Accessed 2 Dec 2017].

Fielding, R., 2007. *Architectural Styles and the Design of Network-based Software Architectures* [online]. Available from: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm [Accessed 2 Dec 2017].

Forbes, 2010. *A New Way To Outsource* [online]. Available from: https://www.forbes.com/2010/06/01/vested-outsourcing-microsoft-intel-leadership-managing-kate-vitasek.html [Accessed 29 Nov 2017].

Fowler, M. and Lewis, J., 2014. *Microservices: a definition of this new architectural term* [online]. Available from: https://martinfowler.com/articles/microservices.html [Accessed 2 Dec 2017].

Gulati, R. and Garino, J., 2000. Get the Right Mix of Bricks and Clicks. *Harvard Business Review*, 78, no. 3 (May-June 2000), 107–114.

Hohpe, G. and Woolf, W., 2004. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA: Addison-Wesley.

IBM, 2010. *Customizing WebSphere Commerce* [online]. Available from: https://www.ibm.com/support/knowledgecenter/en/SSZLC2_8.0.0/com.ibm.commerce.developer.doc/concepts/cdestorecustomizer.htm [Accessed 29 Nov 2017].

IETF, 2014. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing* [online]. Available from: https://tools.ietf.org/html/rfc7230 [Accessed 2 Dec 2017].

Kossak, F., Illibauer, C., Geist, V., Geist, V., Kubovy, J., Natschläger, C., Ziebermayr, T., Kopetzky, T., Freudenthaler, B., and Schewe, K.-D., 2014. *A Rigorous Semantics for BPMN 2.0 Process Diagrams*. Springer.

McCarthy, T., 2017. *Amazon's first New York bookstore blends tradition with technology* [online]. Available from: https://www.theguardian.com/technology/2017/may/26/amazon-new-york-bookstore [Accessed 25 Nov 2017].

Microsoft, 2003. *How RPC Works* [online]. Available from: https://technet.microsoft.com/en-us/library/cc738291(v=ws.10).aspx [Accessed 2 Dec 2017].

mymuesli GmbH, 2017. *Our Story* [online]. Available from: https://uk.mymuesli.com/story [Accessed 18 Nov 2017].

OMG, 2017. *OMG Unified Modeling Language* [online]. Available from: http://www.omg.org/spec/UML/2.5/ [Accessed 20 Nov 2017].

Papazoglou, M. P. and Ribbers, P. M., 2006. *e-Business: Organizational and Technical Foundations*.

Chichester: John Wiley and Sons.

PCI, 2017. *PCI Security Standards* [online]. Available from: https://www.pcisecuritystandards.org/ [Accessed 2 Dec 2017].

Pine, B. J., II, Victor, B., and Boynton, A. C., 1993. Making Mass Customization Work. *Harvard Business Review*, 71, no. 5 (September-October 1993), 108–119.

Porter, M. E., 2008. The Five Competitive Forces That Shape Strategy. *Harvard Business Review*, 86, no. 1 (January 2008), 25–40.

Rappa, M., 2010. *Business Models on the Web* [online]. Available from: http://digitalenterprise.org/models/models.html [Accessed 25 Nov 2017].

SmartBear, 2017. *OpenAPI Specification, Version 3.0.0* [online]. Available from: https://swagger.io/specification/ [Accessed 2 Dec 2017].

Stair, R. M. and Reynolds, G. W., 2014. *Principles of Information Systems*. 11th Edition. Boston, MA: Cengage Learning.

The Economist, 2009. *Just-in-time* [online]. Available from: http://www.economist.com/node/13976392 [Accessed 2 Dec 2017].

Timmers, P., 1998. Business Models for Electronic Markets. *Electronic Markets*, 8 (2), 3–8.

Tseng, M. M. and Jiao, J., 2001. Mass Customization. *In*: Salvendy, G., ed. *Handbook of Industrial Engineering: Technology and Operations Management*. New York, NY: John Wiley and Sons, 684–709.

van der Aalst, W. M. and Stahl, C., 2011. *Modeling Business Processes: A Petri Net-Oriented Approach*. Cambridge, MA: MIT Press.

W3C, 2007. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)* [online]. Available from: https://www.w3.org/TR/soap12/ [Accessed 2 Dec 2017].

W3C, 2012. *Media Queries* [online]. Available from: https://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619/ [Accessed 2 Dec 2017].

Yip, P., 2005. *Basic Concepts of Co-Branding, With Examples from the Hospitality Industry Could Co-branding Improve Your Bottomline?* [online]. Available from: http://www.hotel-online.com/News/PR2005_3rd/Sep05_CoBranding.html [Accessed 25 Nov 2017].

Zeppelin GmbH, 2016. *ReciPure: The Mix Factory of the Future* [online]. Available from: https://www.iso.org/standard/62021.html [Accessed 18 Nov 2017].